6CS005 Learning Journal - Prashant Piya

**Contents**

[**1.0 POSIX Threads** 3](#_Toc29113588)

[**1.1 Password Cracking** 3](#_Toc29113589)

[**1.2 Imagine Processing** 11](#_Toc29113590)

[**1.3 Linear Regression** 24](#_Toc29113591)

[**2.0 Cuda** 34](#_Toc29113592)

[**2.1 Password Cracking** 34](#_Toc29113593)

[**2.2 Image Processing** 38](#_Toc29113594)

[**2.3 Linear Regression** 46](#_Toc29113595)

[**3.0 Message Passing Interface (MPI)** 53](#_Toc29113596)

[**3.1 Password Cracking** 53](#_Toc29113597)

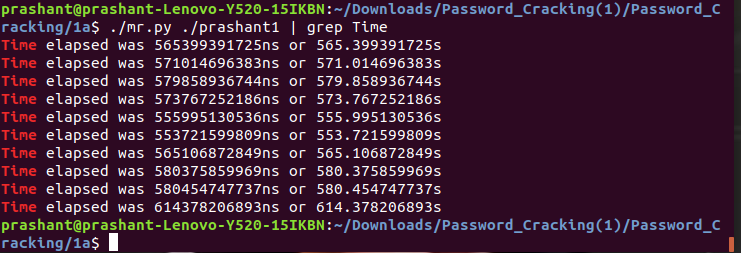
[**3.2 Image Processing** 58](#_Toc29113598)

[**3.3 Linear Regression** 66](#_Toc29113599)

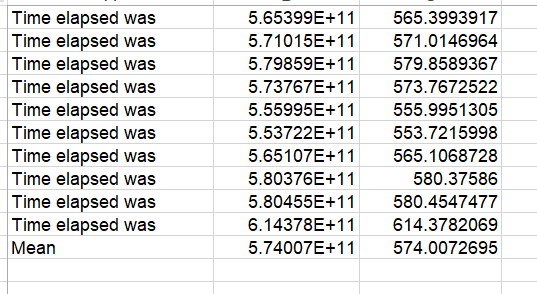
# **1.0 POSIX Threads**

## **1.1 Password Cracking**

Executing the program 10 times.

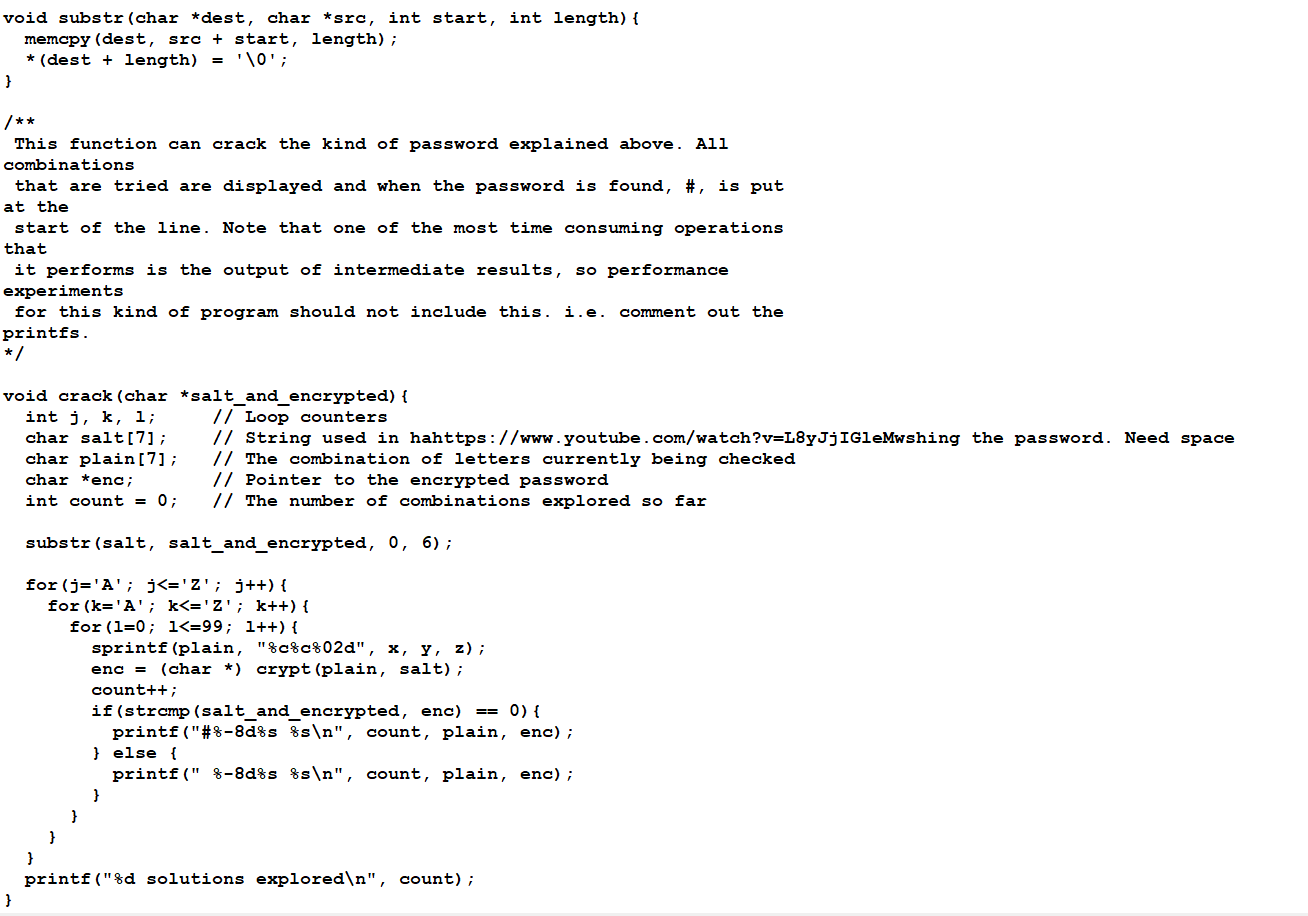


Calculating the mean run time of the above program.



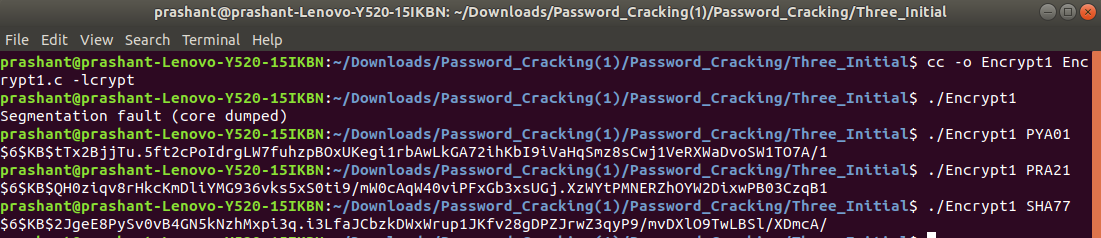
**Source Code:**

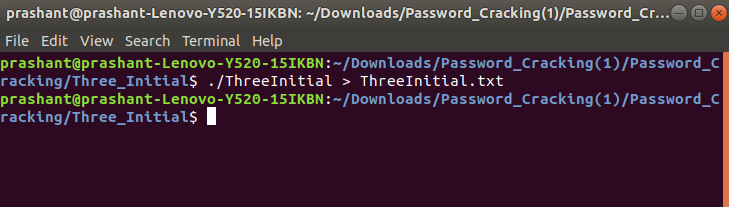






Adding password to the program



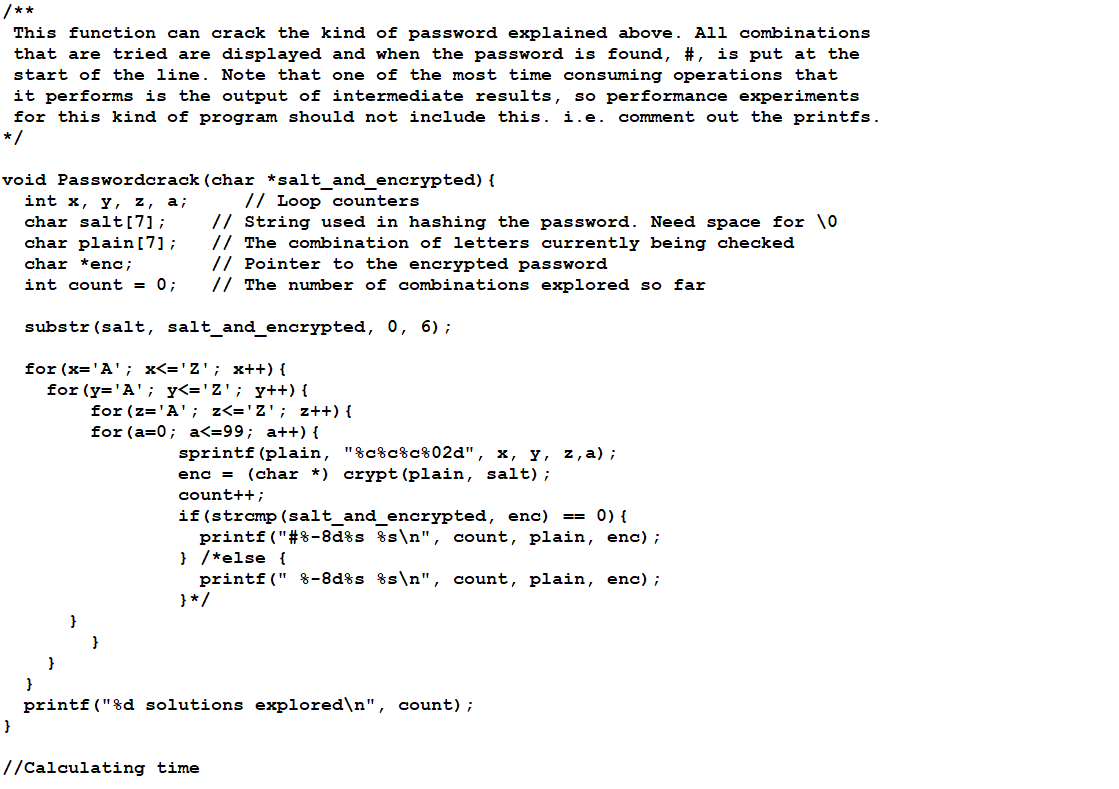
  
The program which cracks two initials password was executed for 10 times and the mean run time was 574.0072 second.

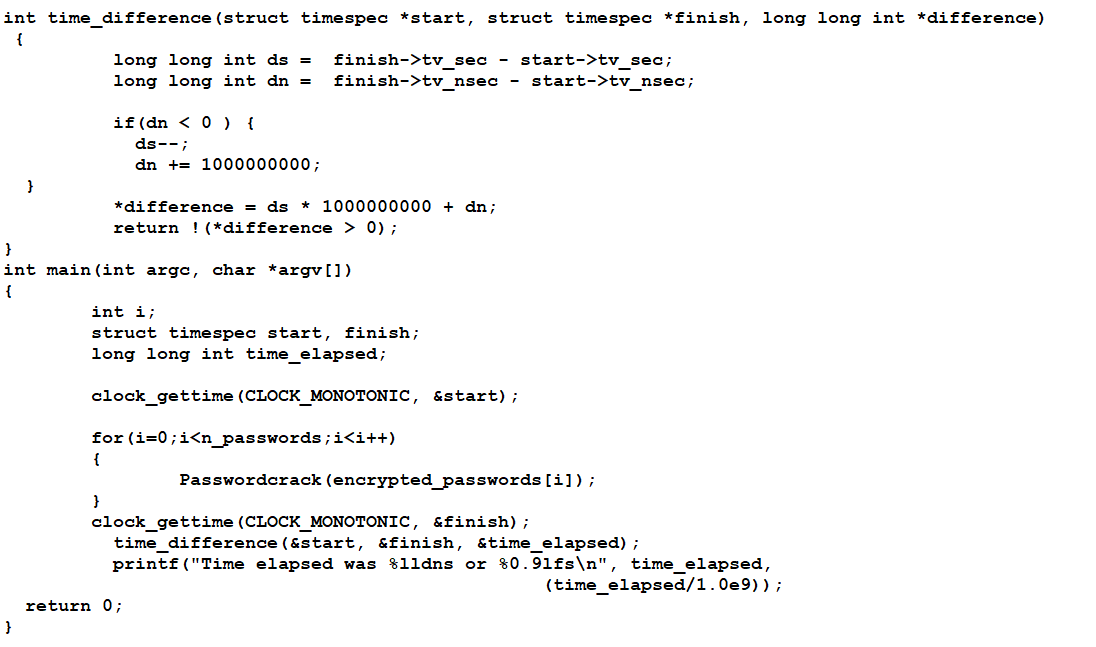
We cracked 3 alphabets and 2 numbers in this program. The password was encrypted using SHA-512 algorithm.

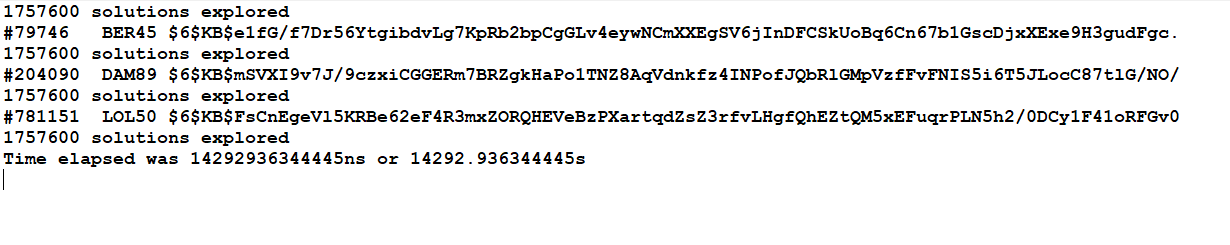
It would take longer than two initials if the number of initials is increased to 3. It's clear that two initial programs take approximately 9 minutes, but if the initial is increased, it's evident that it will take more time to complete a program for loops. If the original is raised to 3 it would take 26 times longer that means it will take about 574\*26 =14,924 second time for completing 3 initial program.

Source code









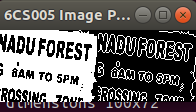
In my hypothesis I assumed it will take about 14,924 second to complete 3 initial program but it took about 14,292 second.

Estimated time=14,924 second. (4.14 hrs.)

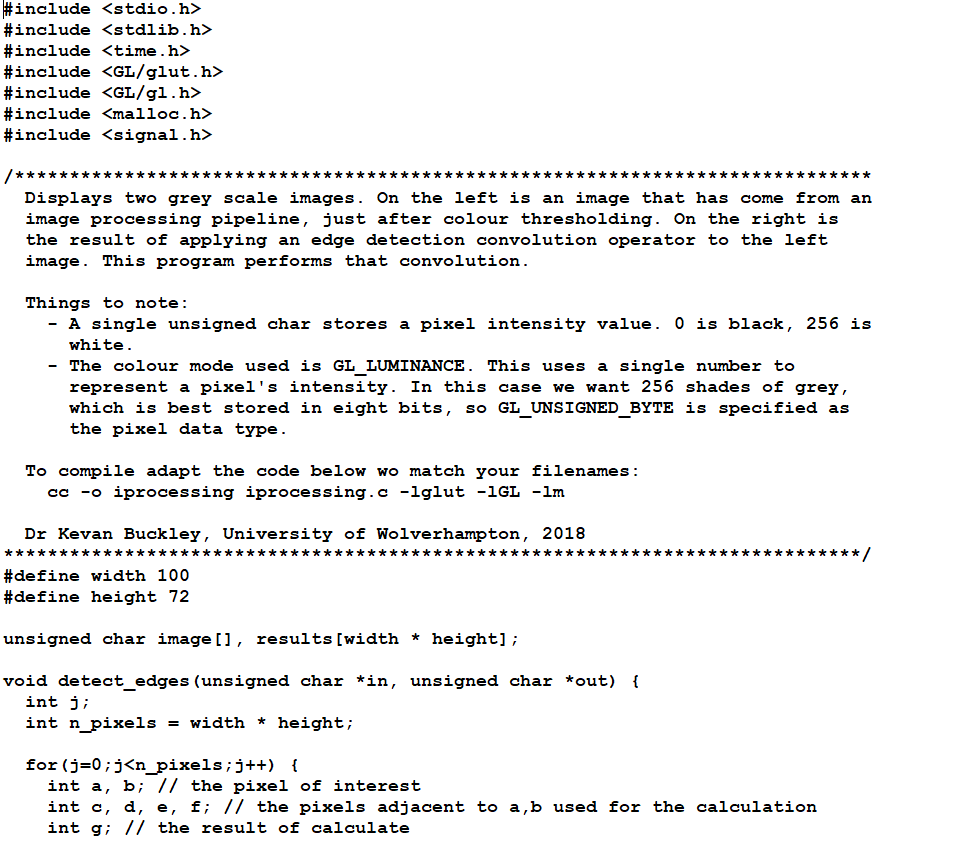
Time difference = 14,924 -14,292 =632 second (10.5 min)

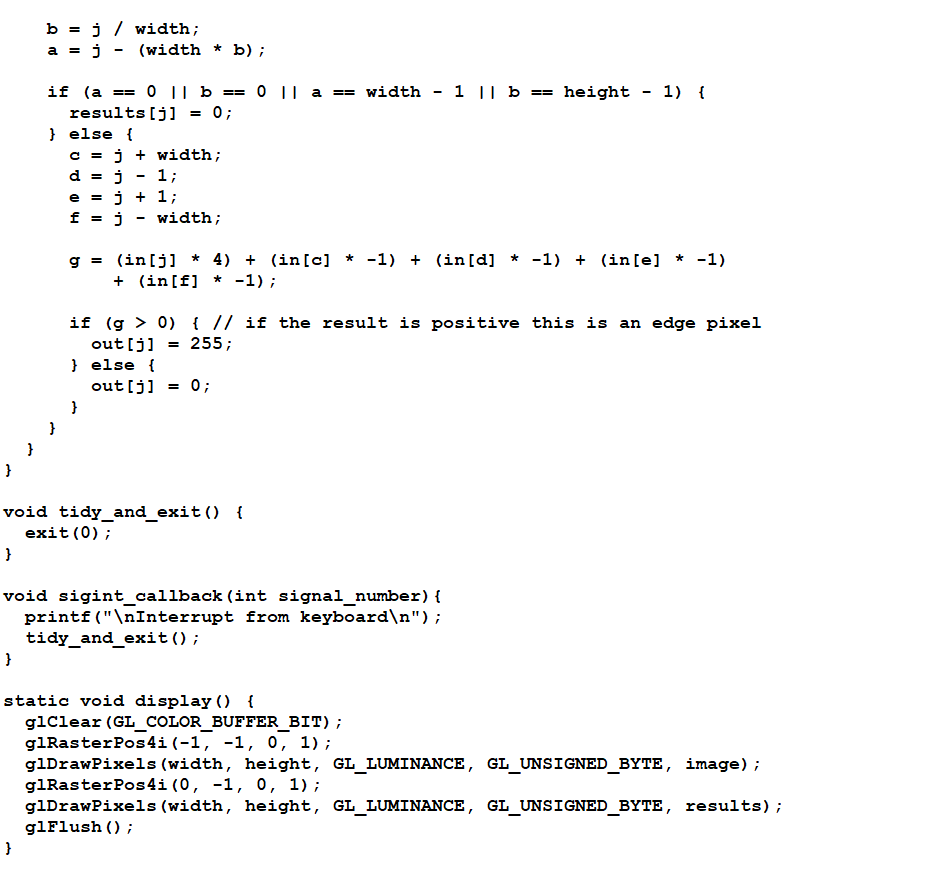
## **1.2 Imagine Processing**

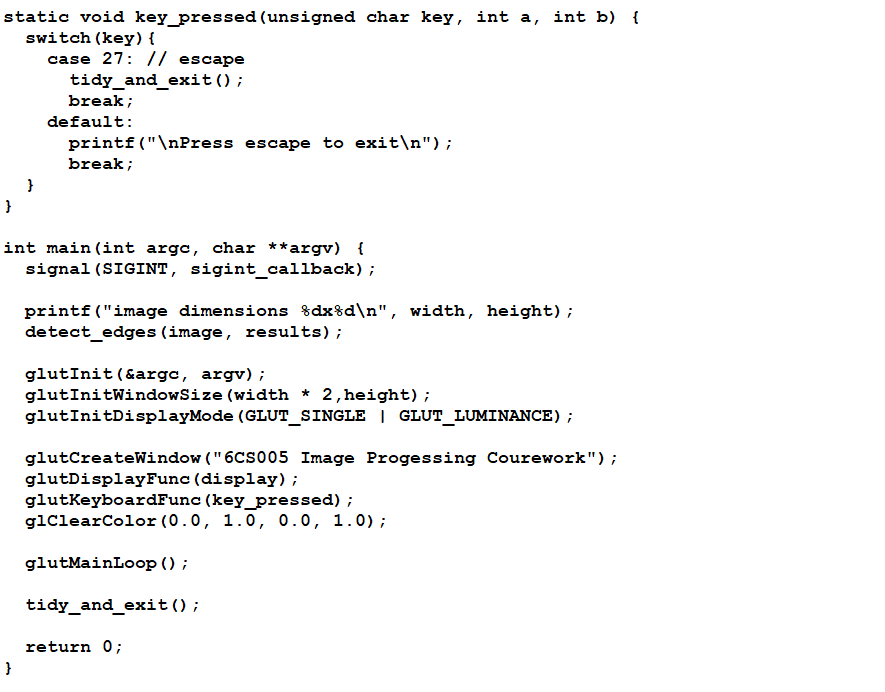
Result after execution the program

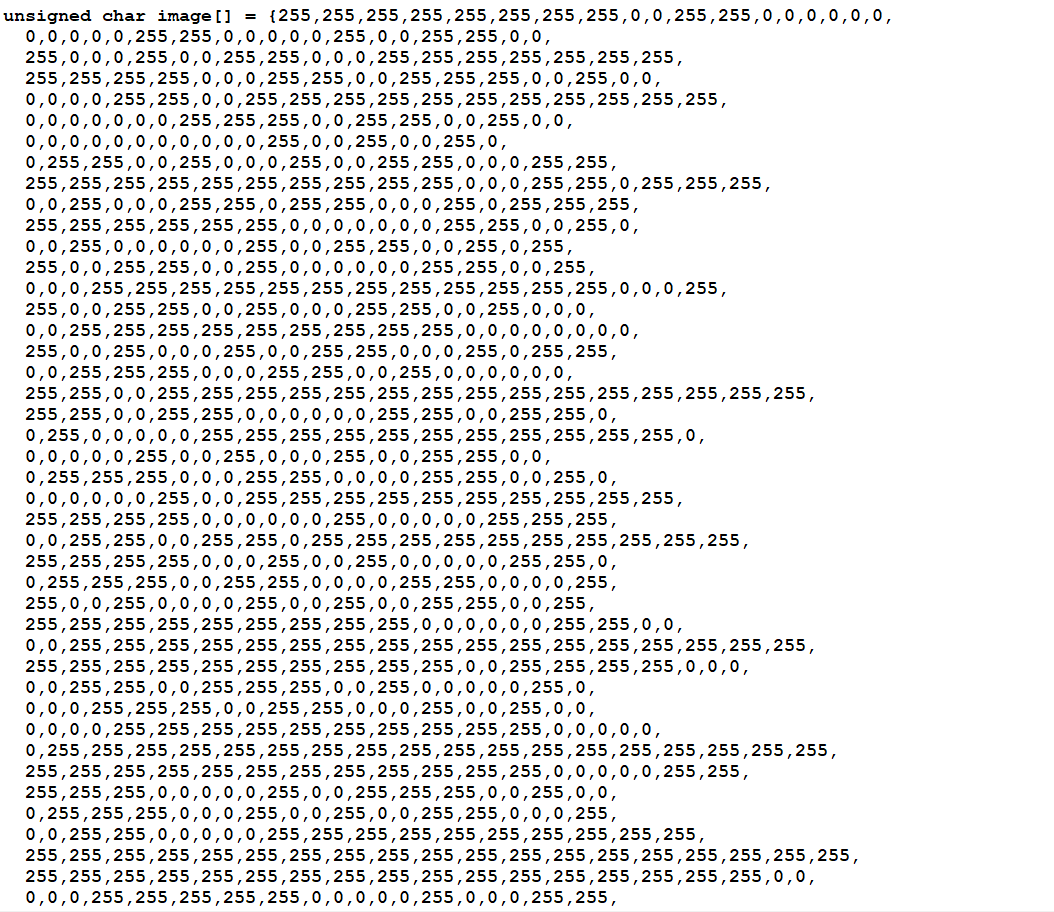


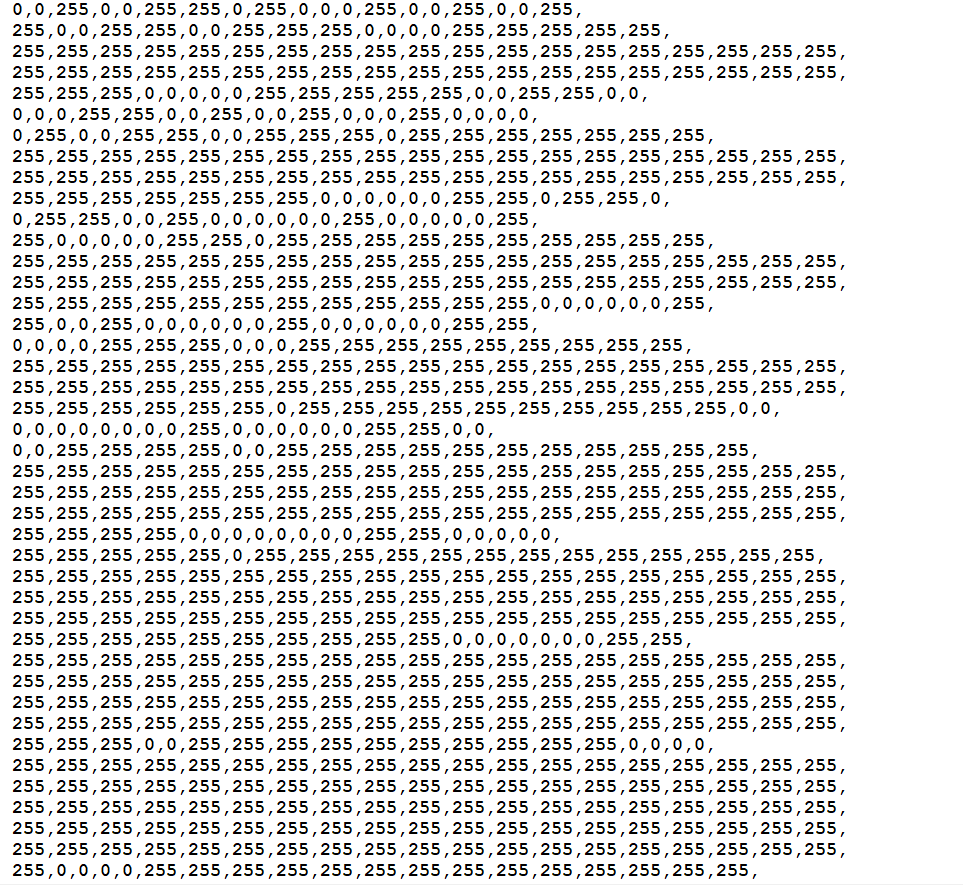
Source code:

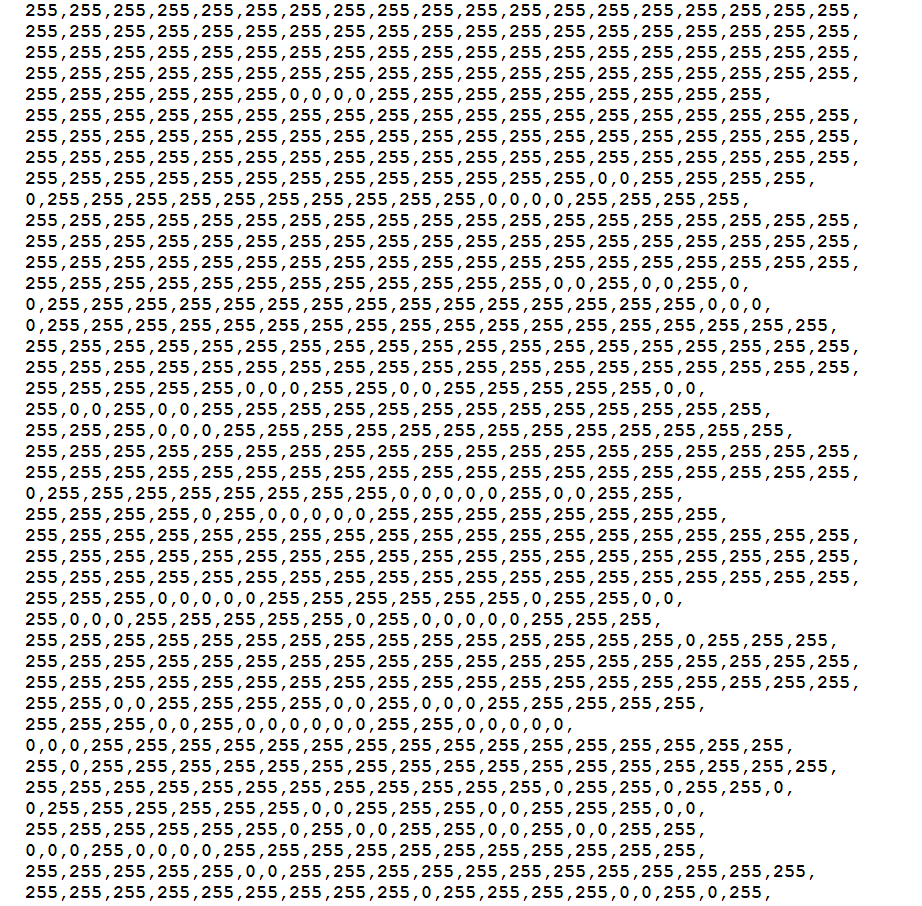


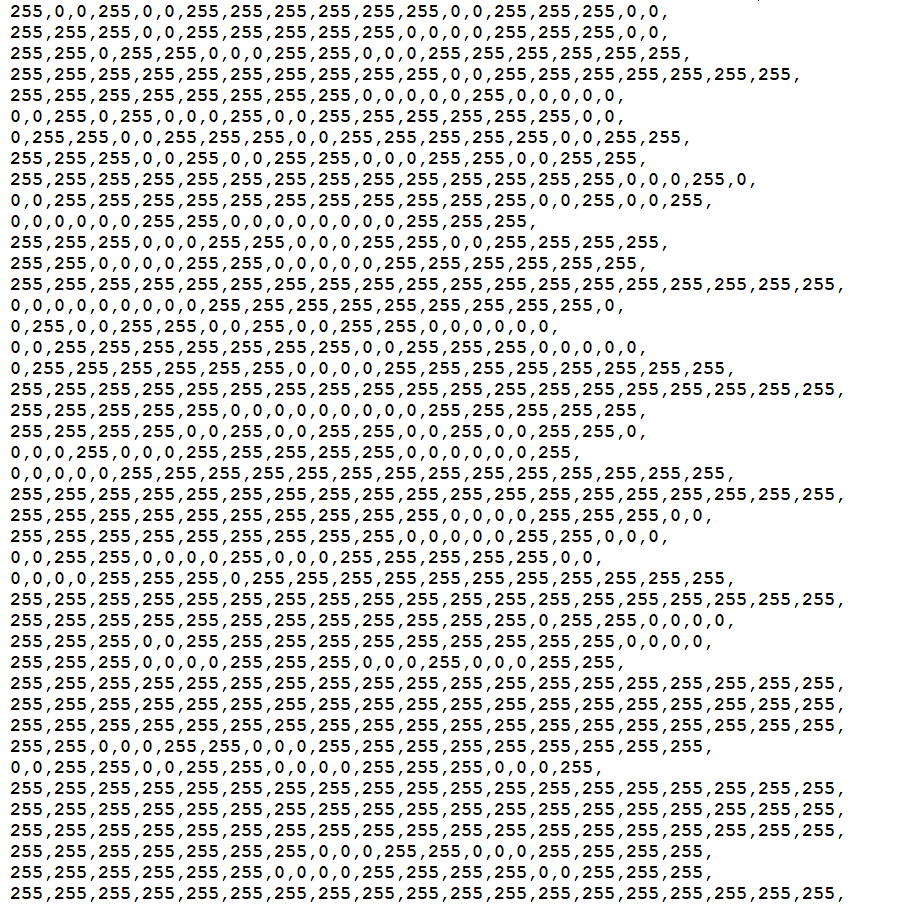


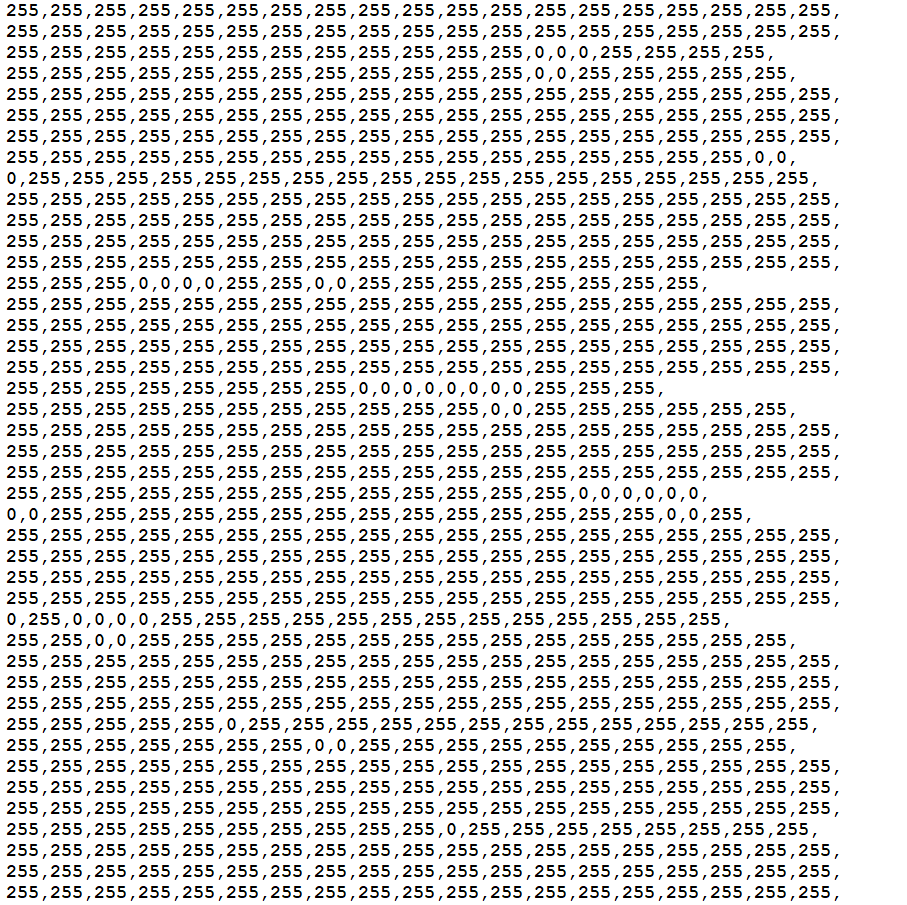


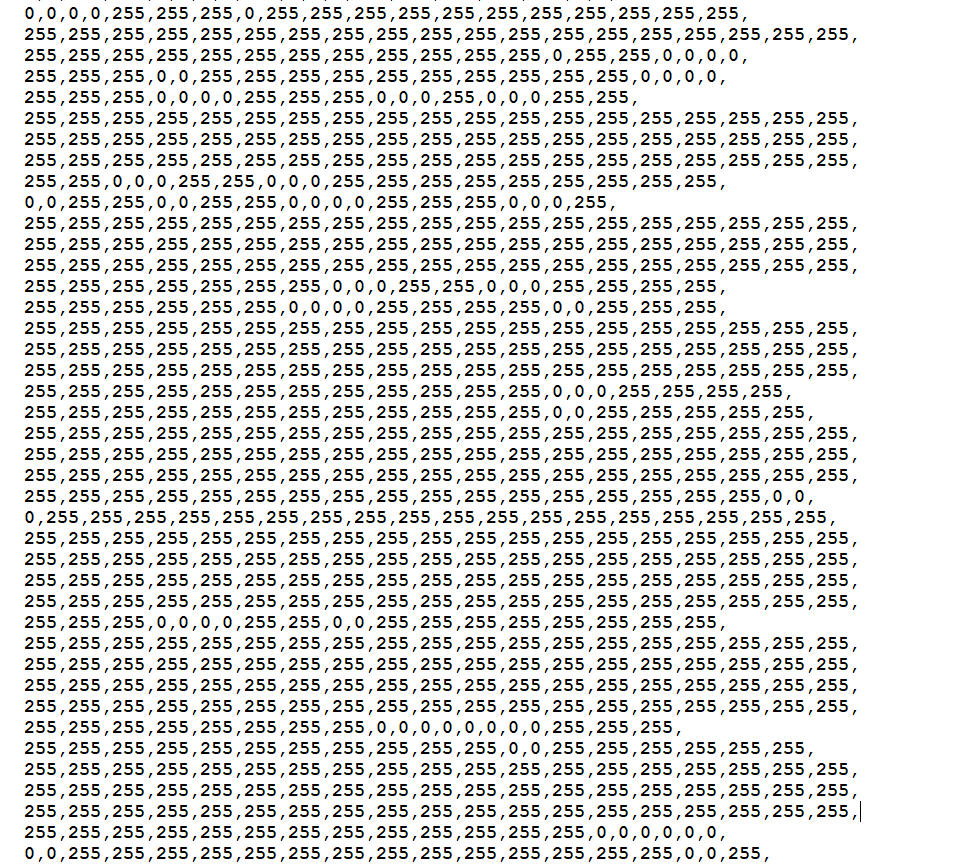


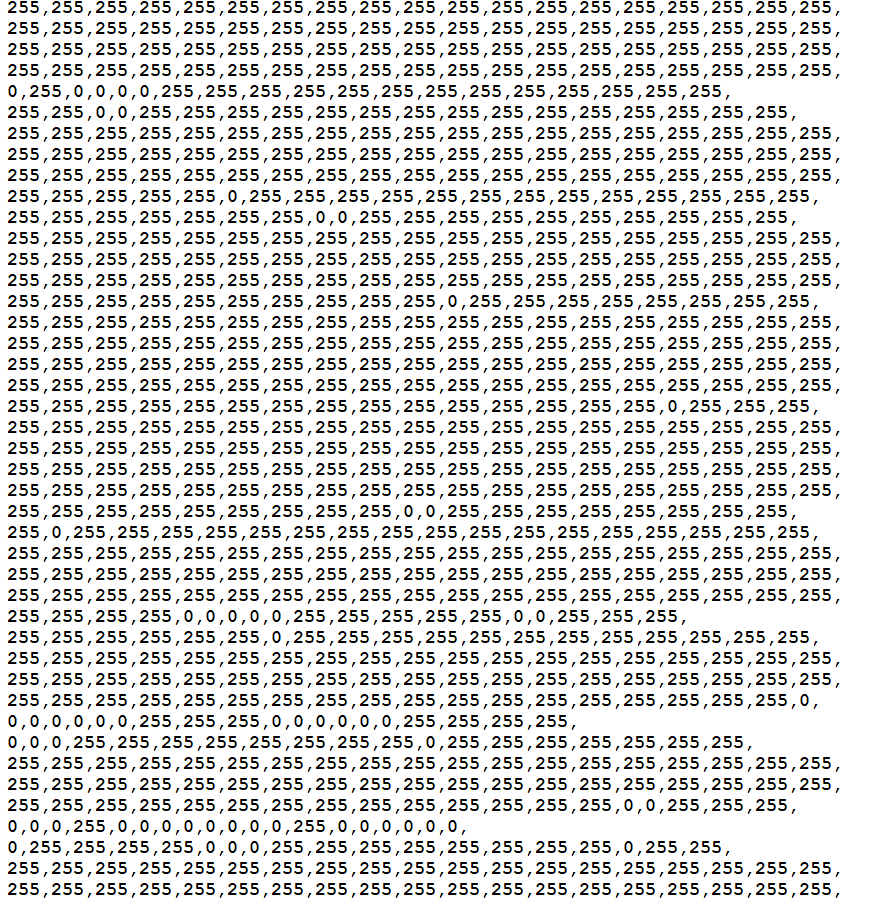


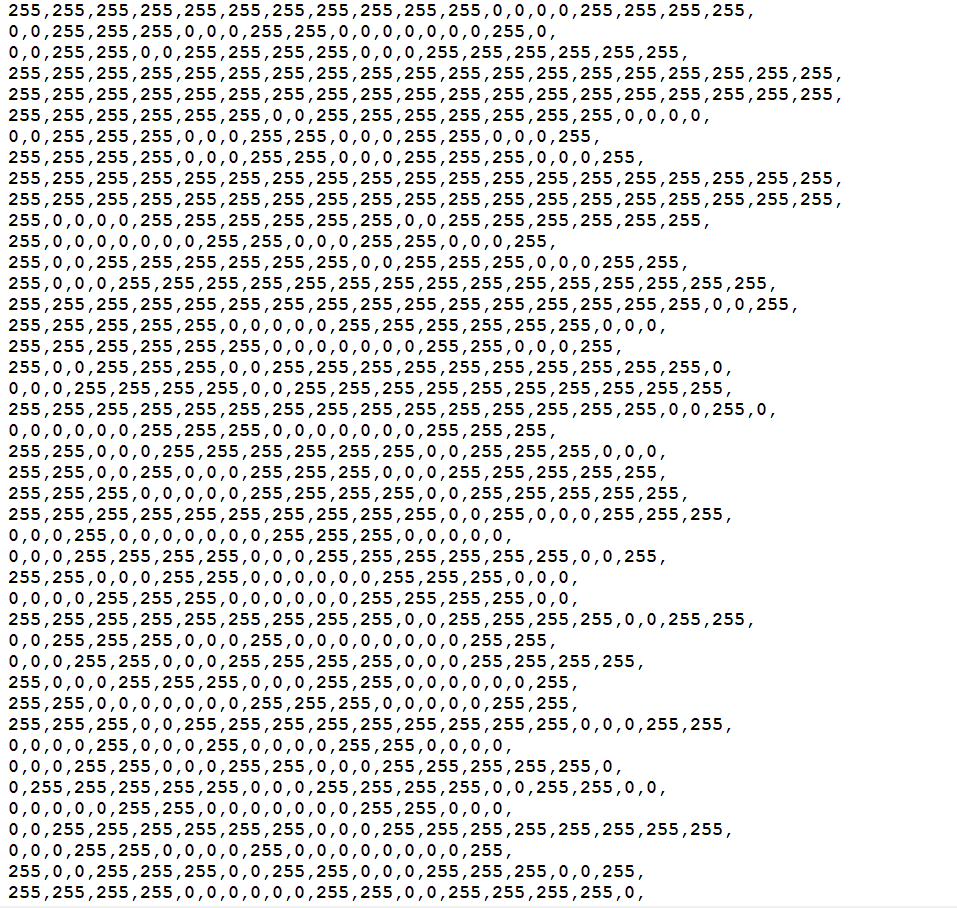


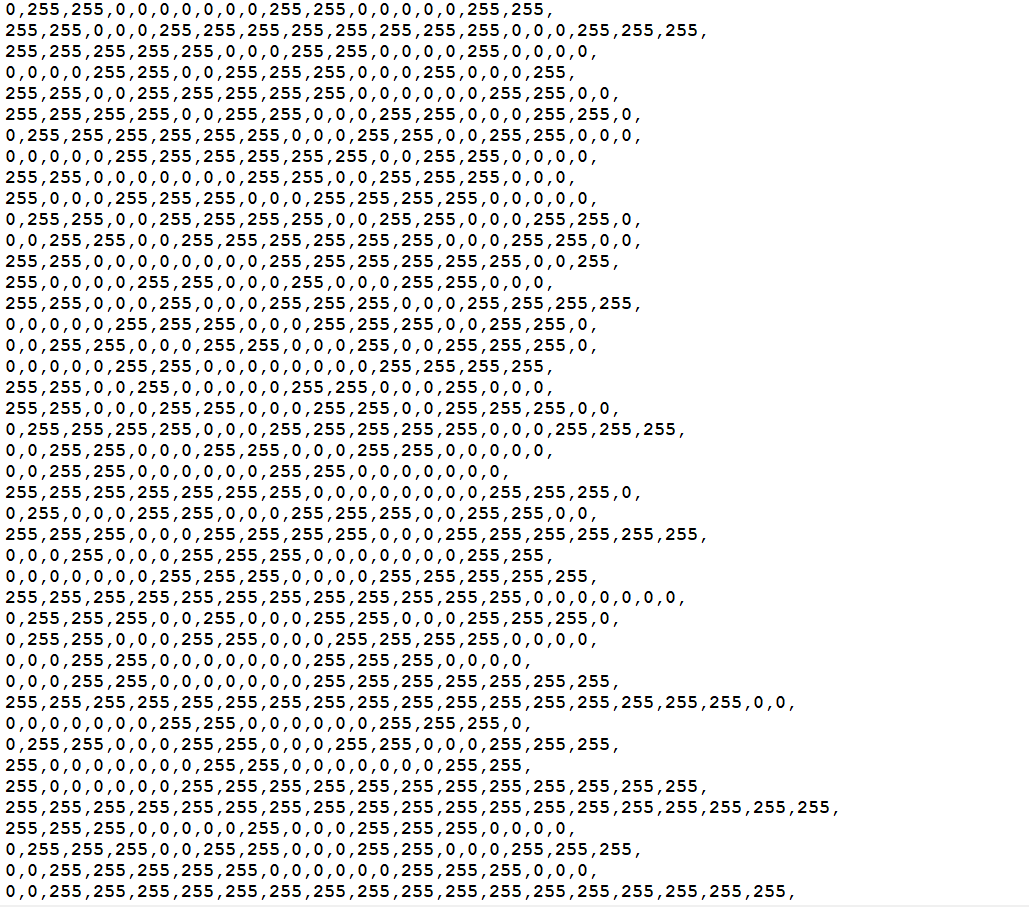


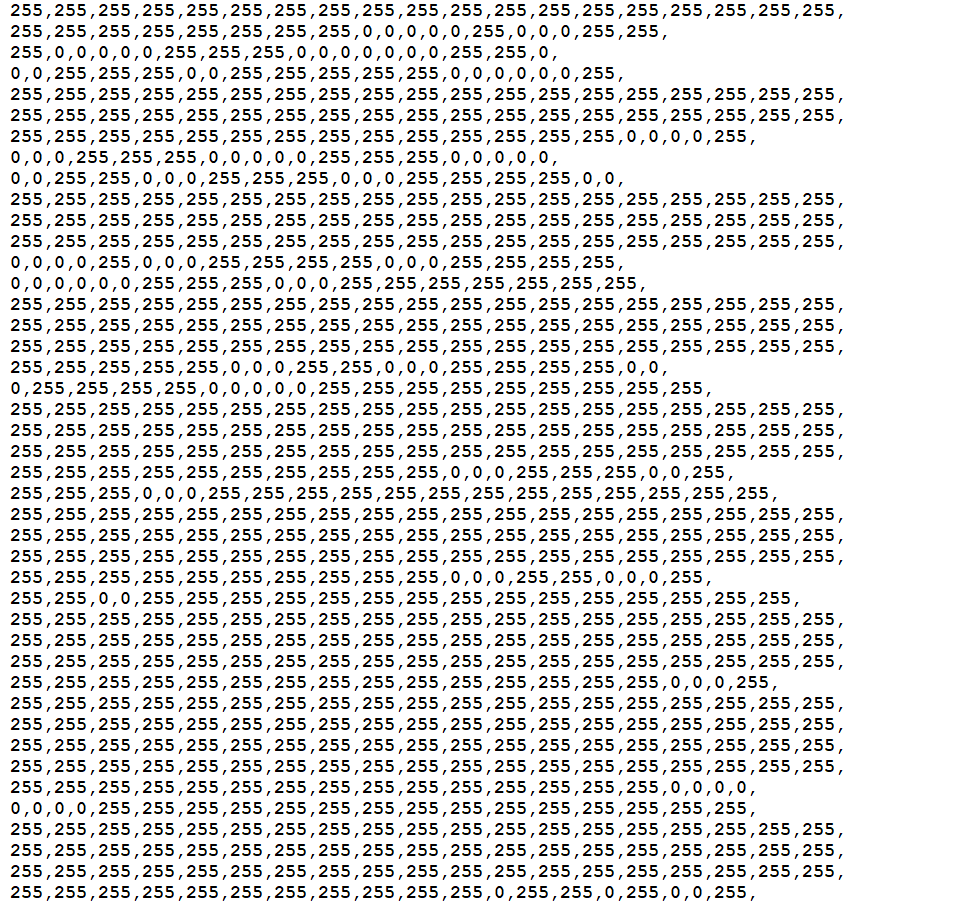












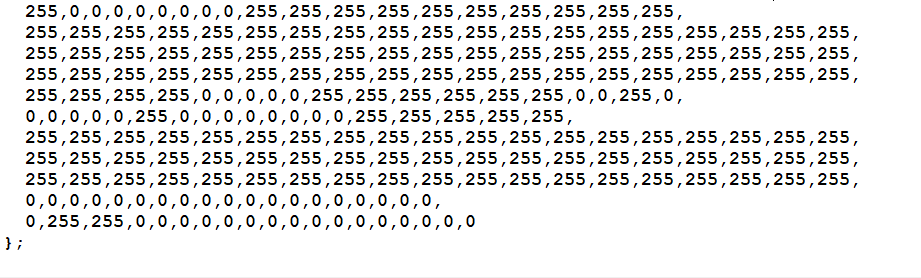
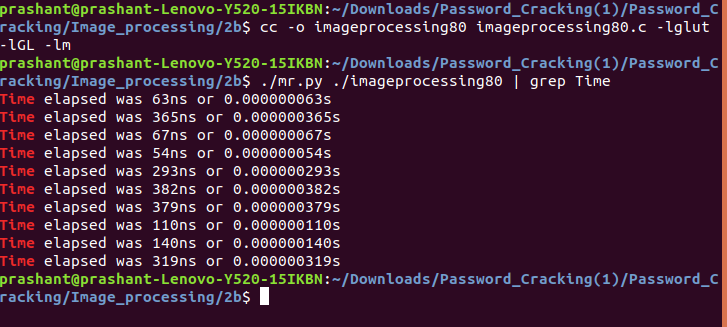
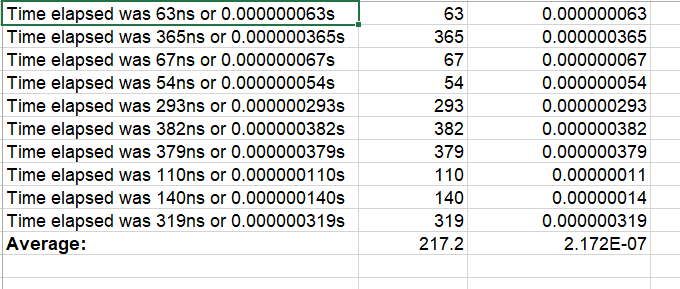


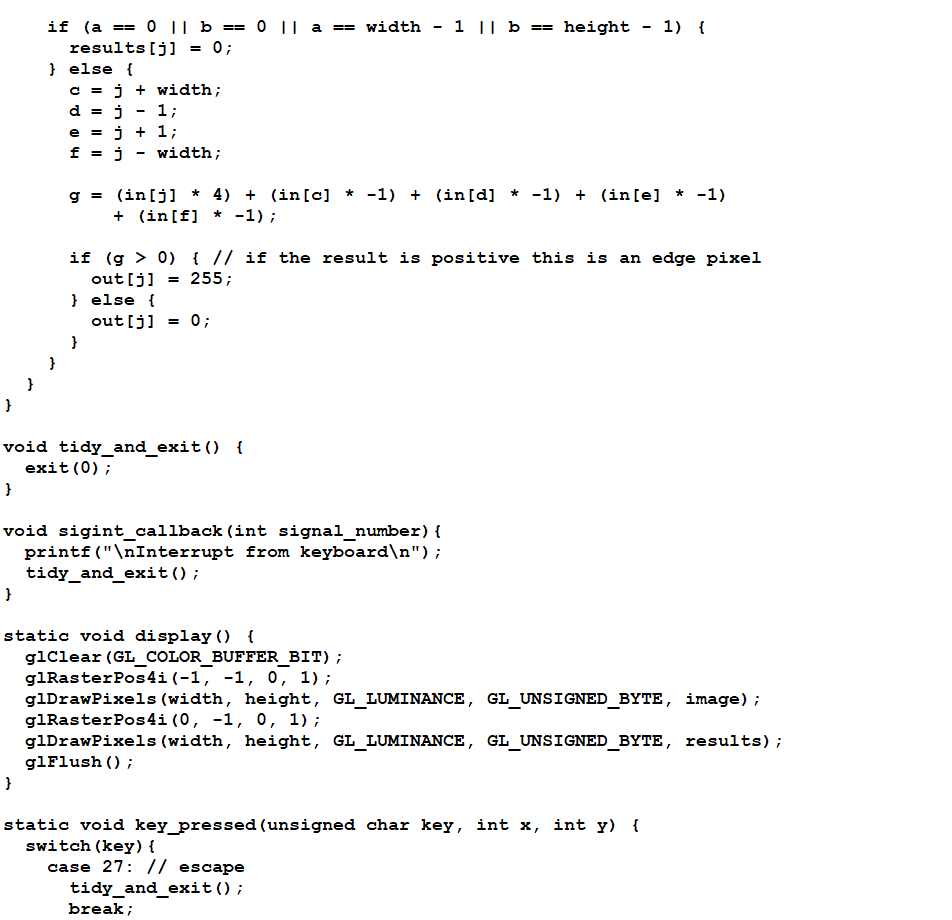
Table with columns that include runtime for the original program and multi-thread version

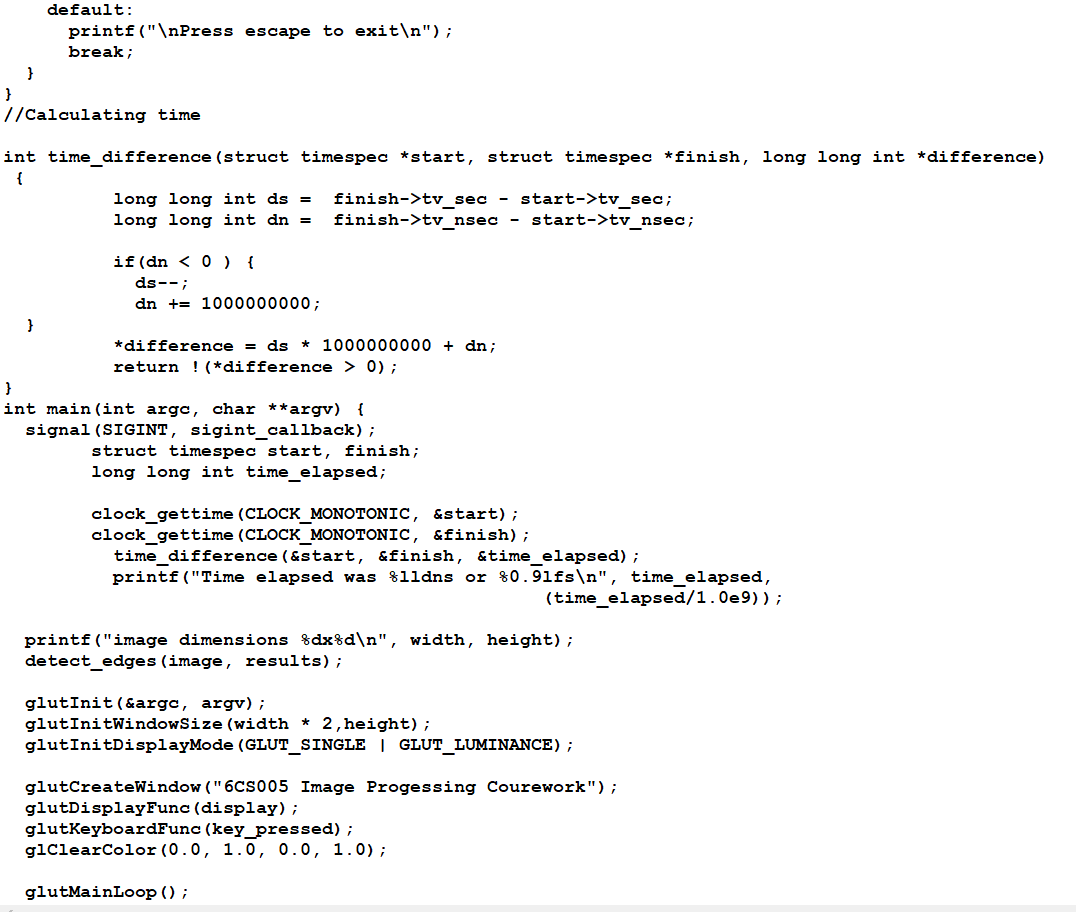


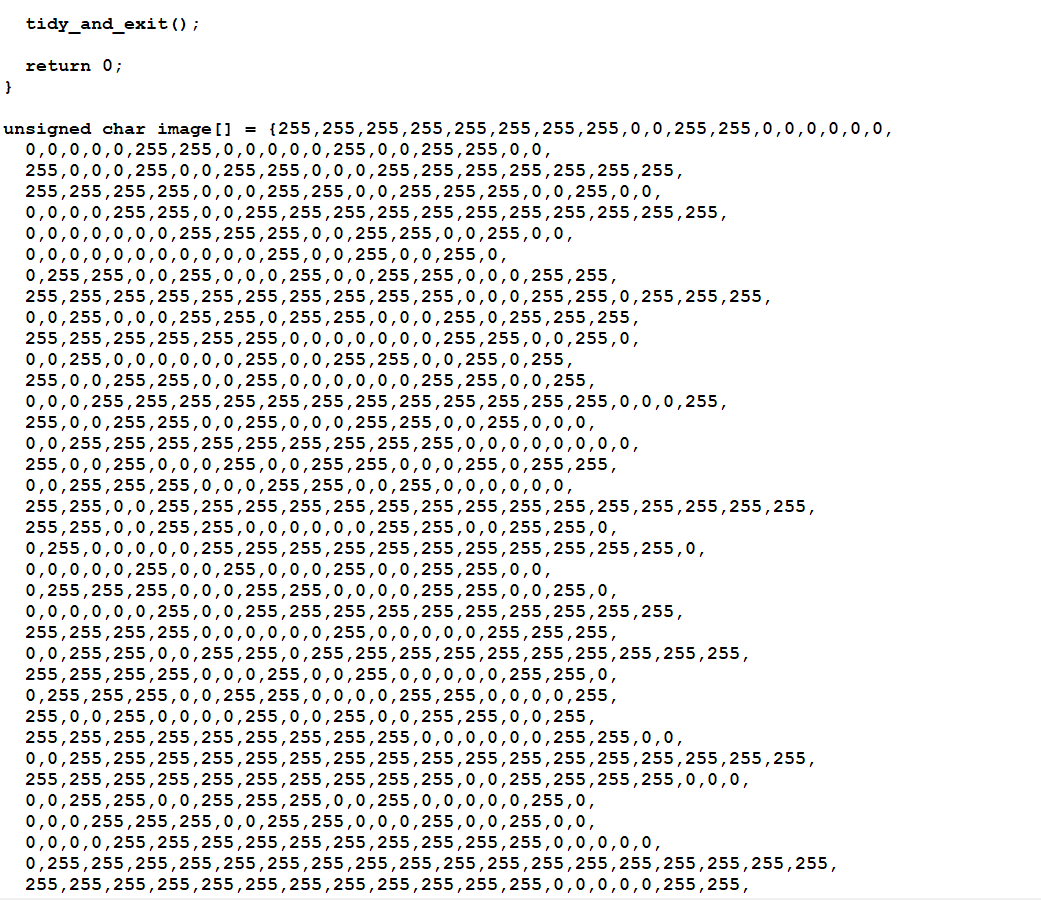


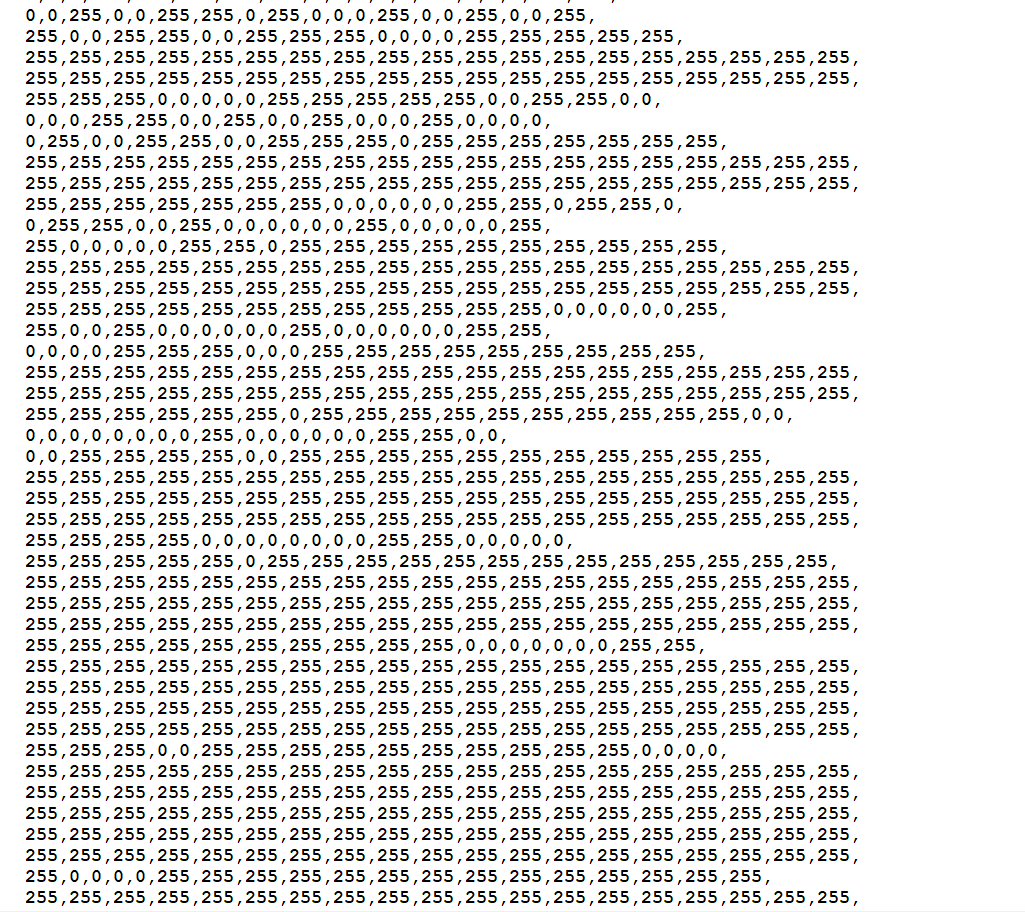
Source code:





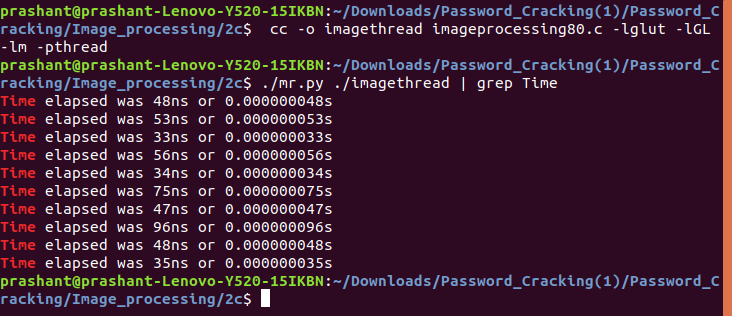


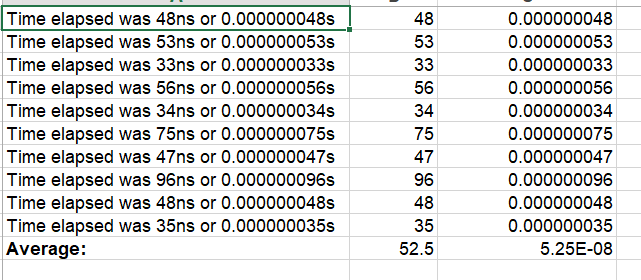




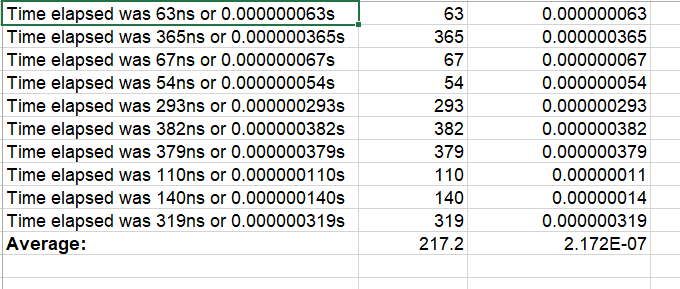
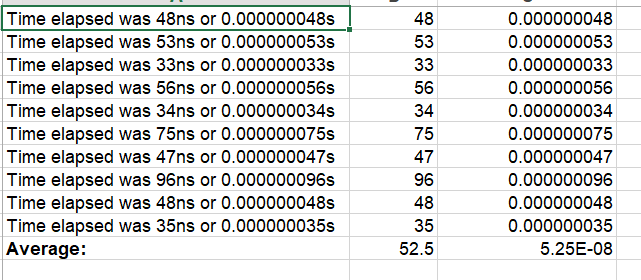


Comparing the runtimes of the original edge detection to the multi thread.





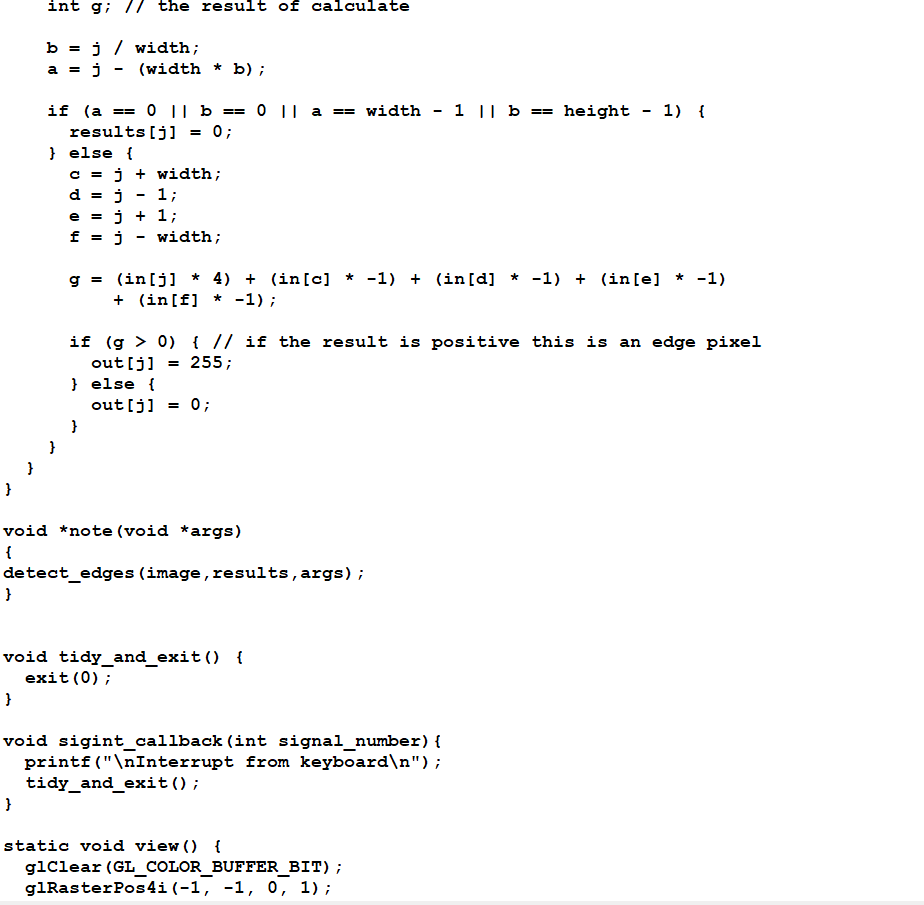
Adding a table that has columns for the original program and multi thread edition that include runtime. Mean running times at the bottom of the columns should be added.



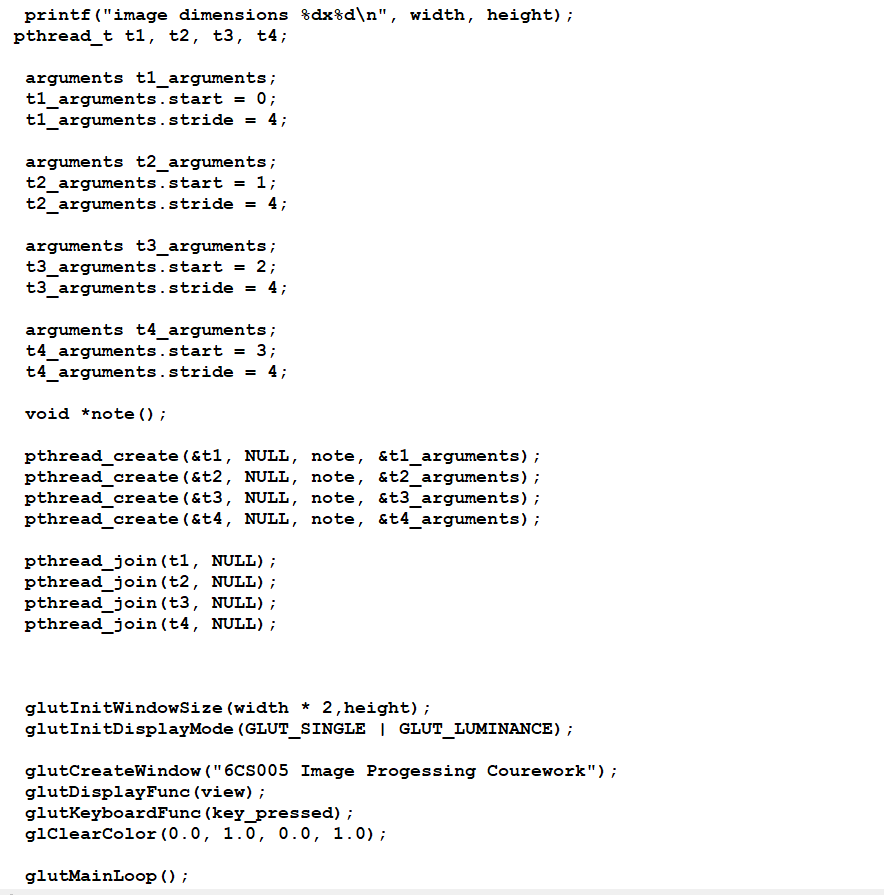
The table above shows the average runtime of the image processing system for multi-thread. It is obvious that the multi-threading software takes less time than a simple program. The explanation is that the multi-threading method increases performance and competitiveness.

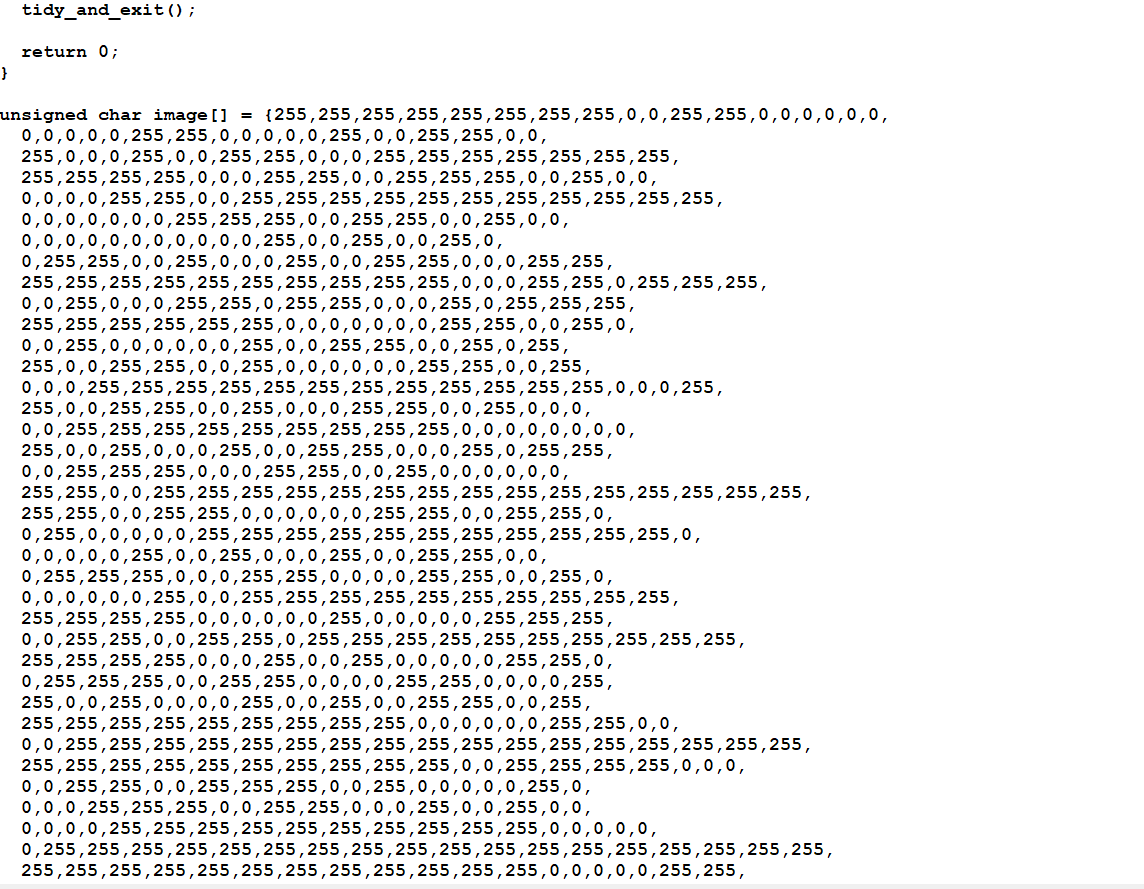
Source code:

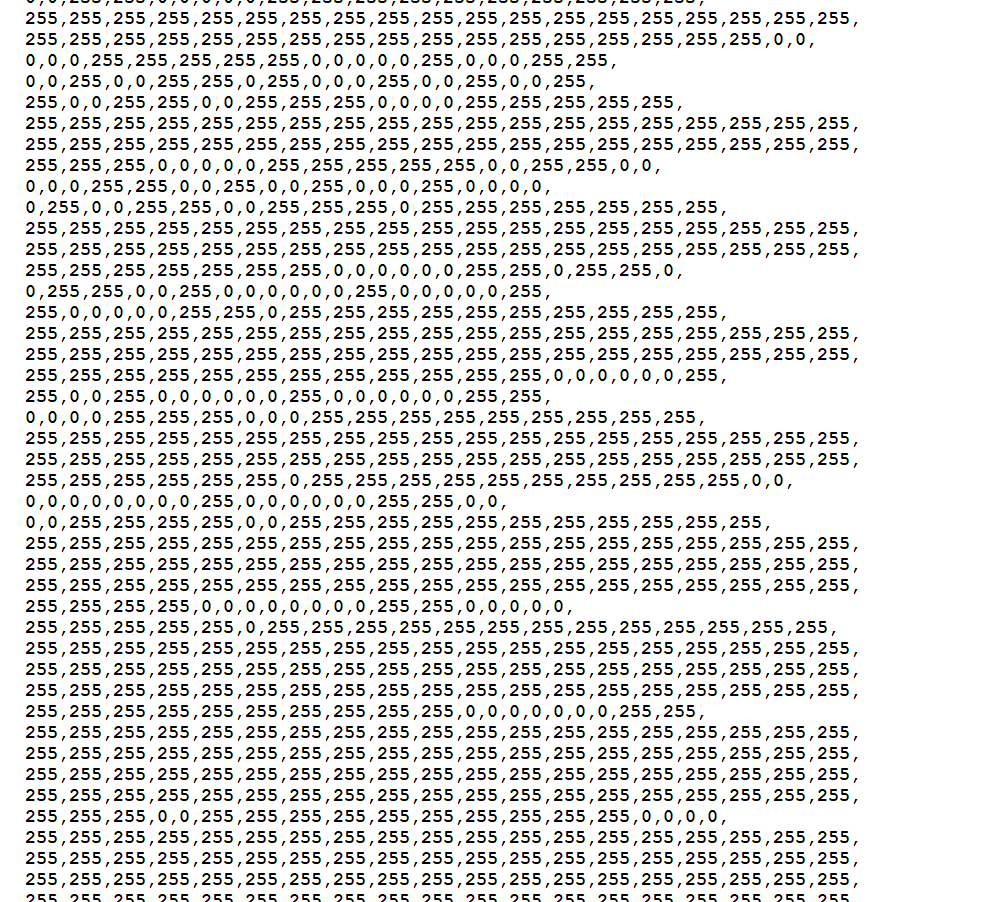


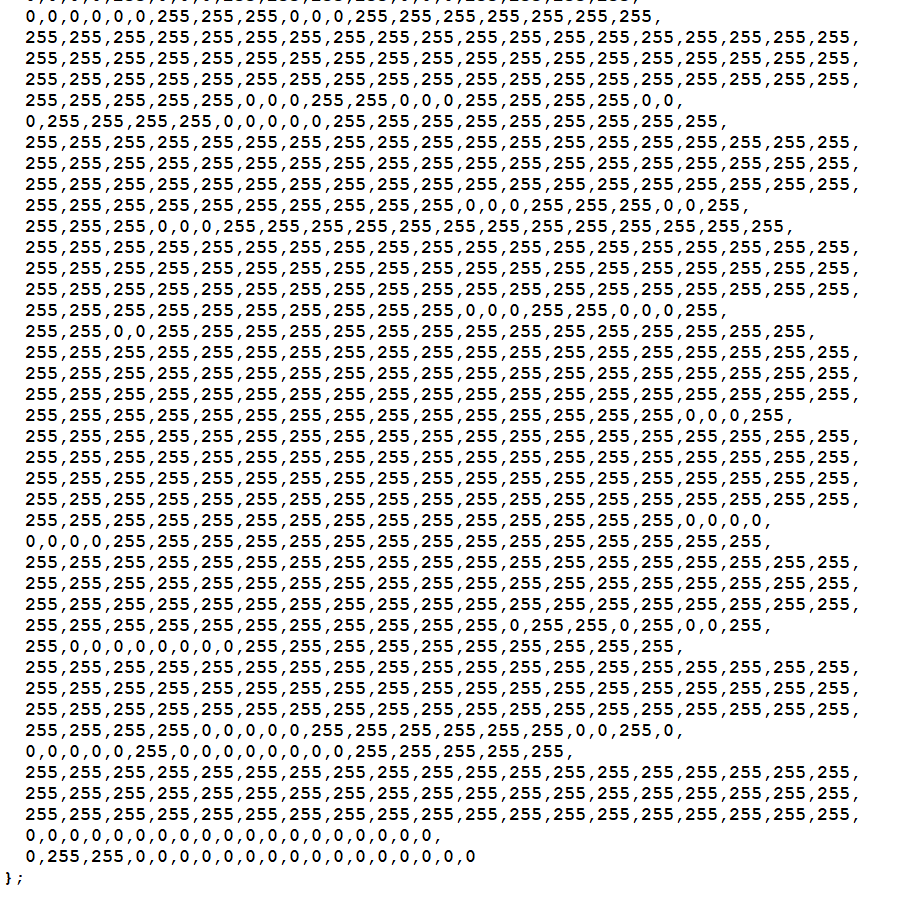






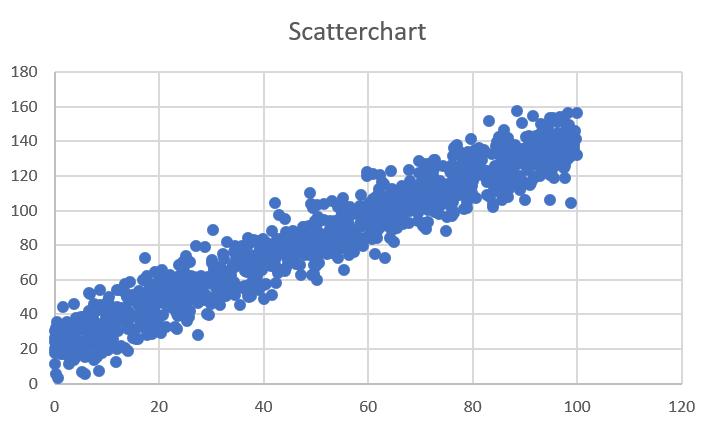


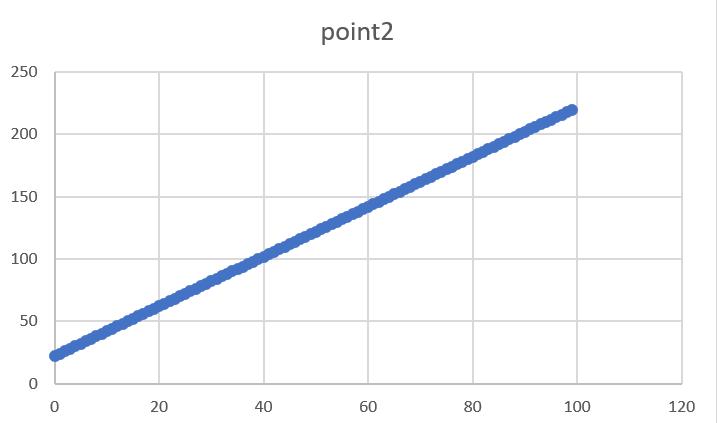


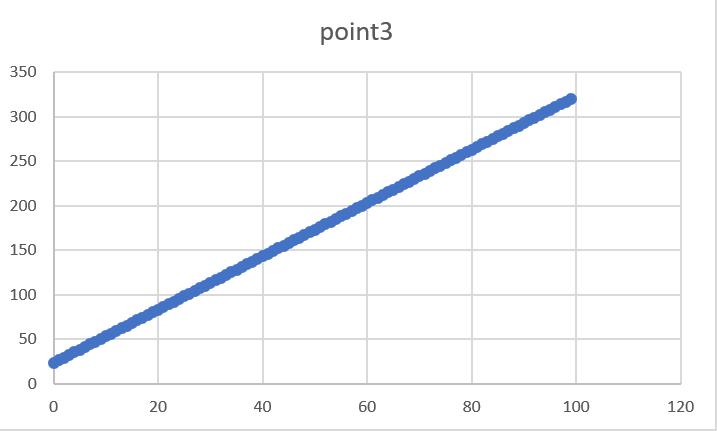


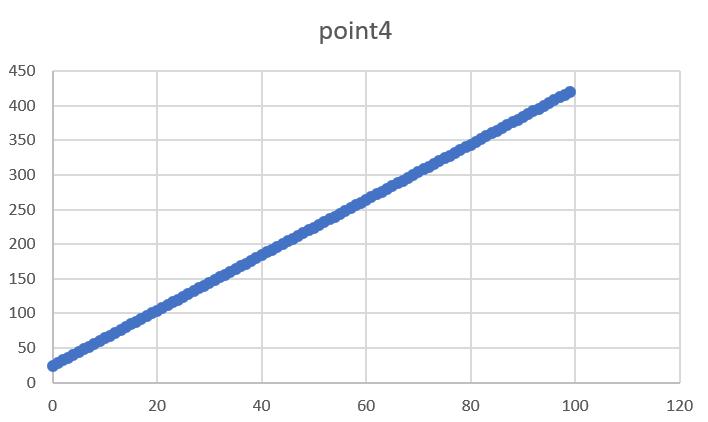
## **1.3 Linear Regression**

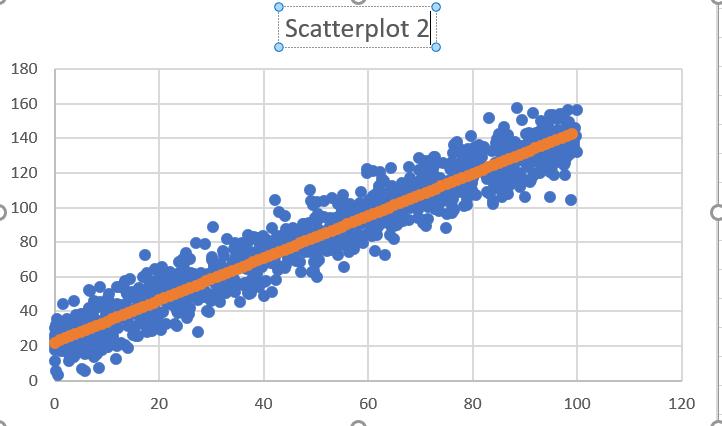
Scatter plot of portfolio

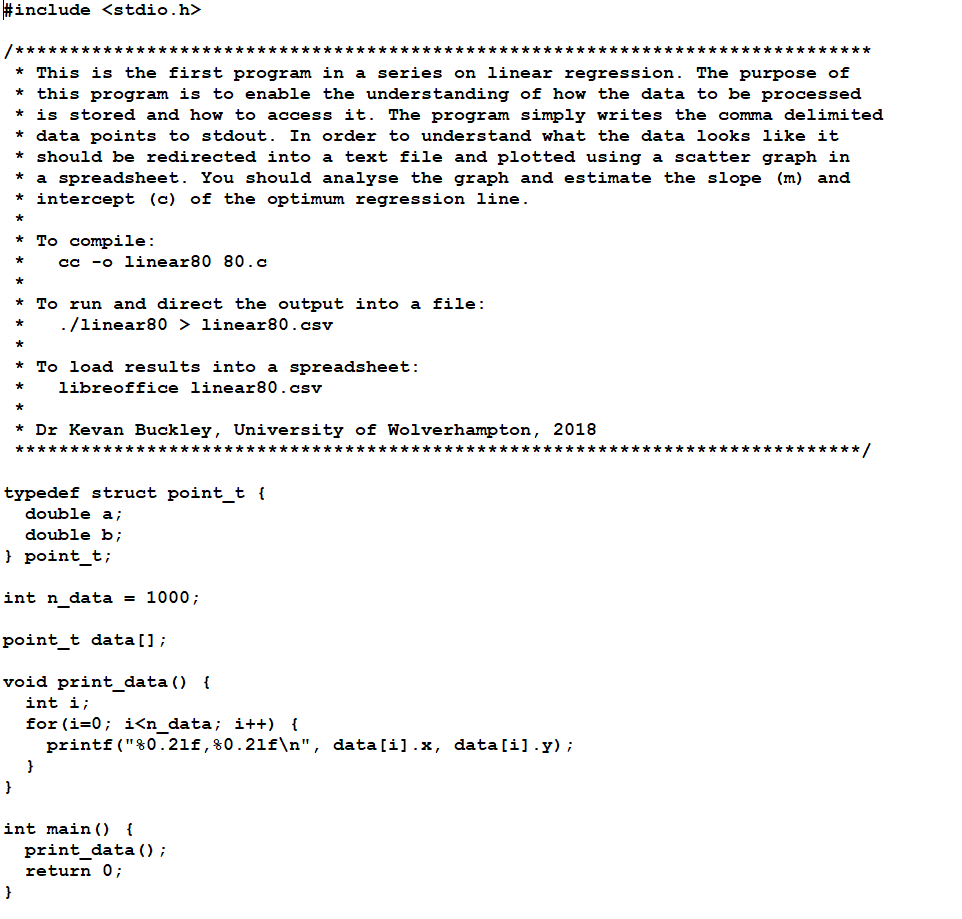


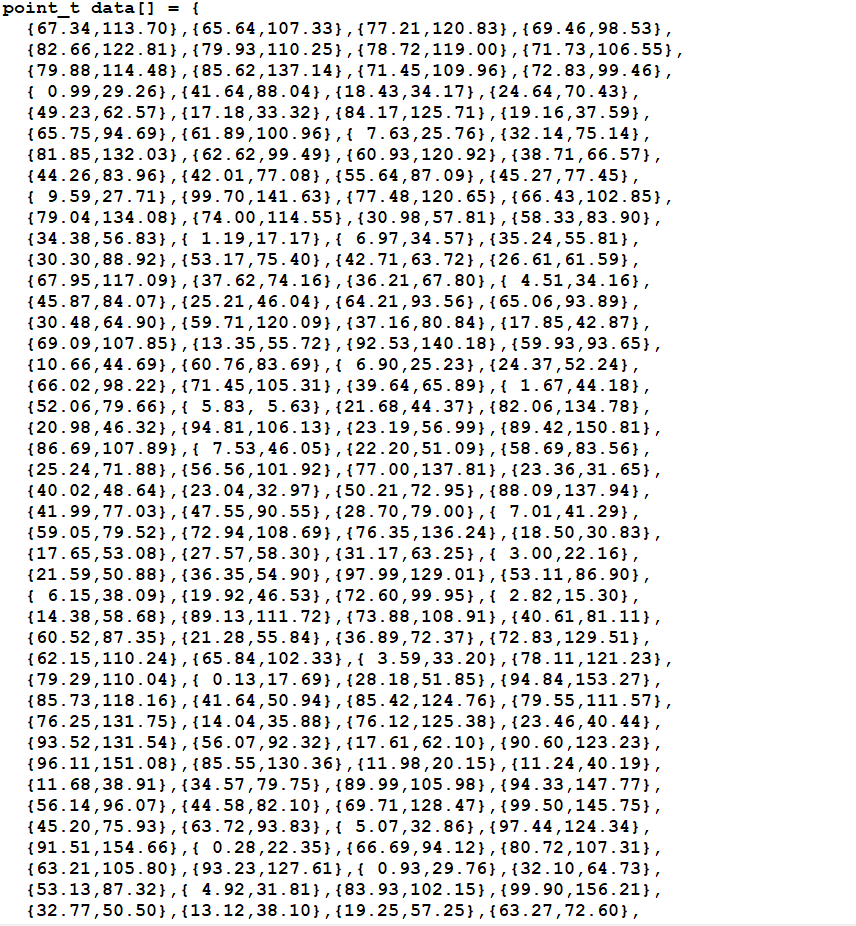


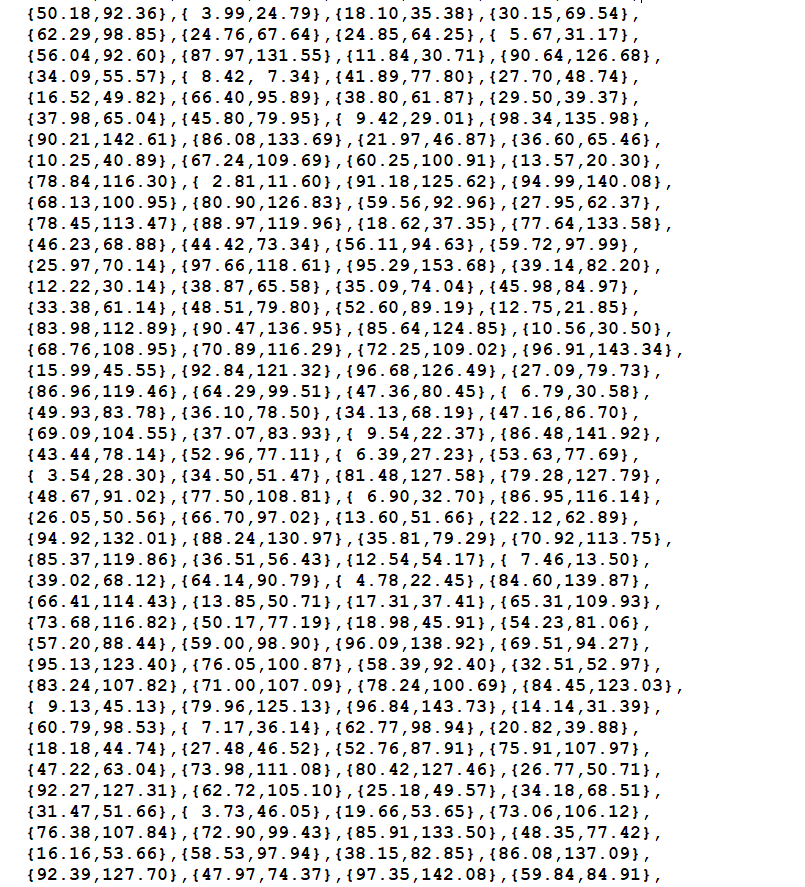


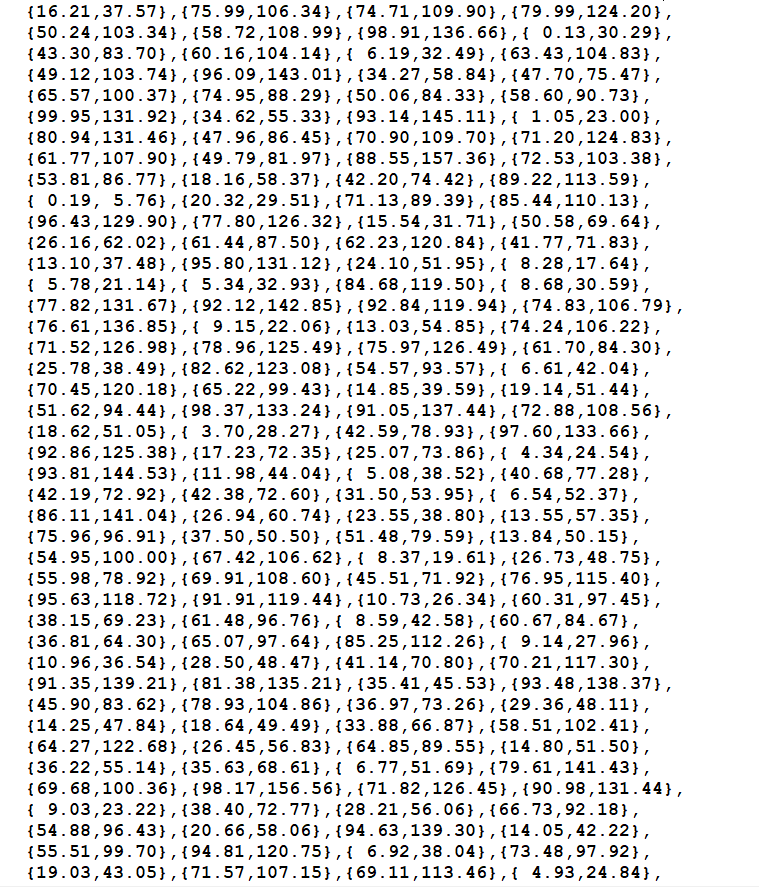


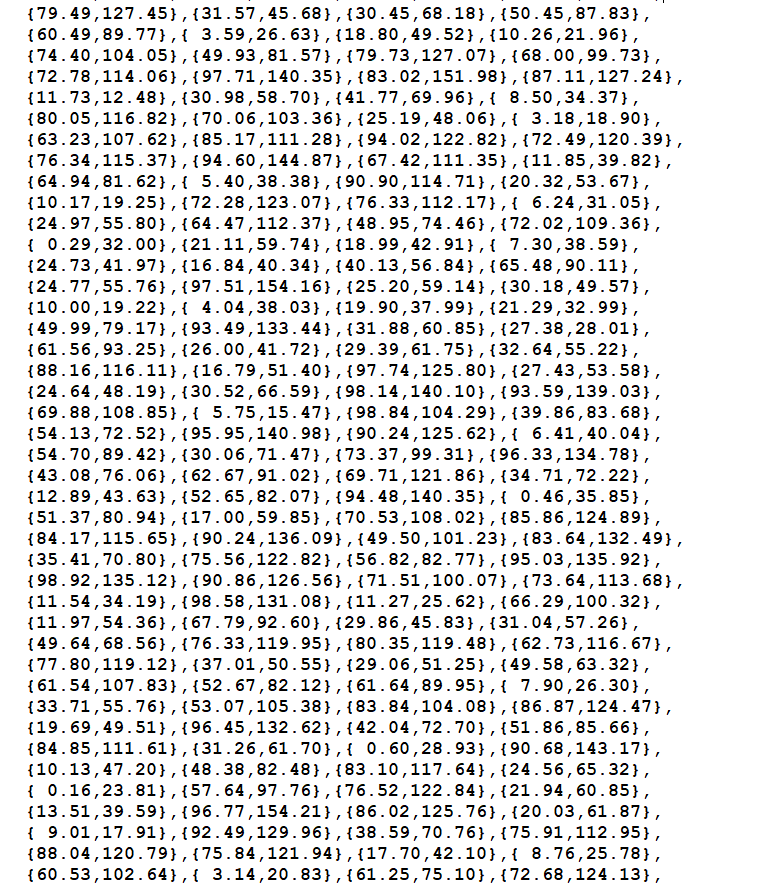


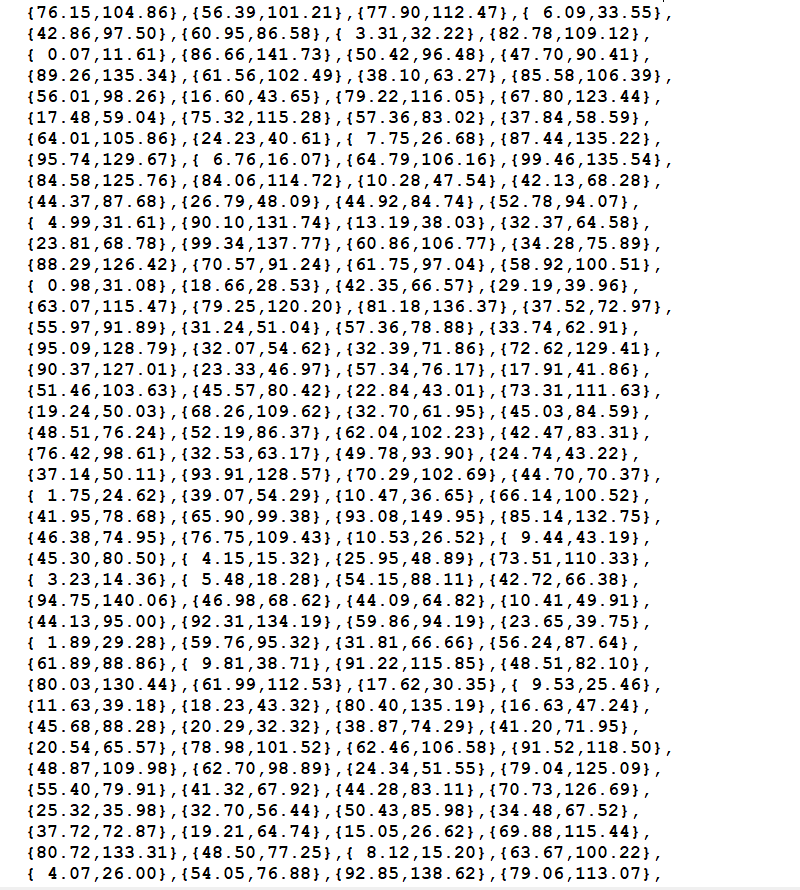


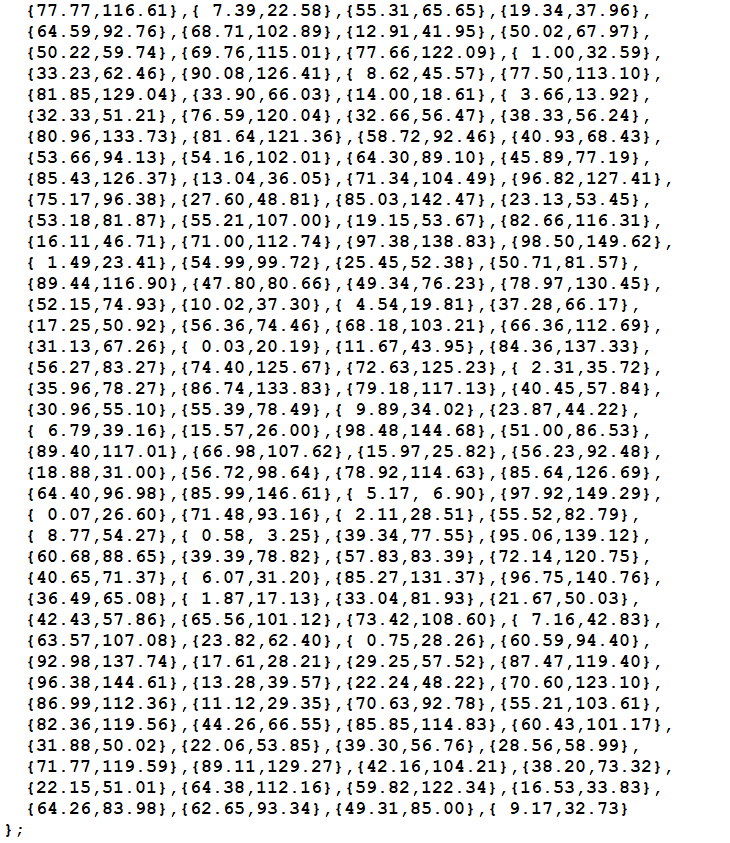


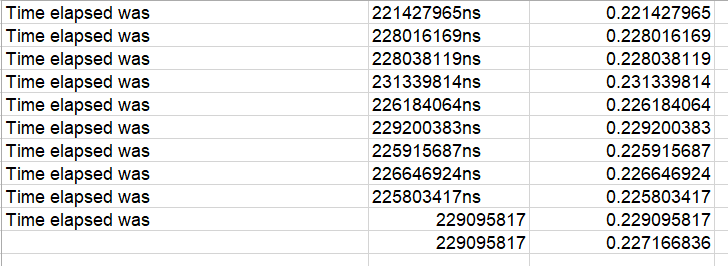








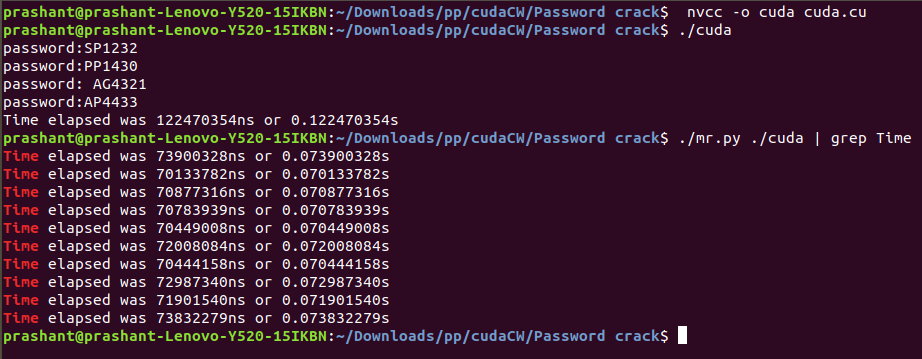




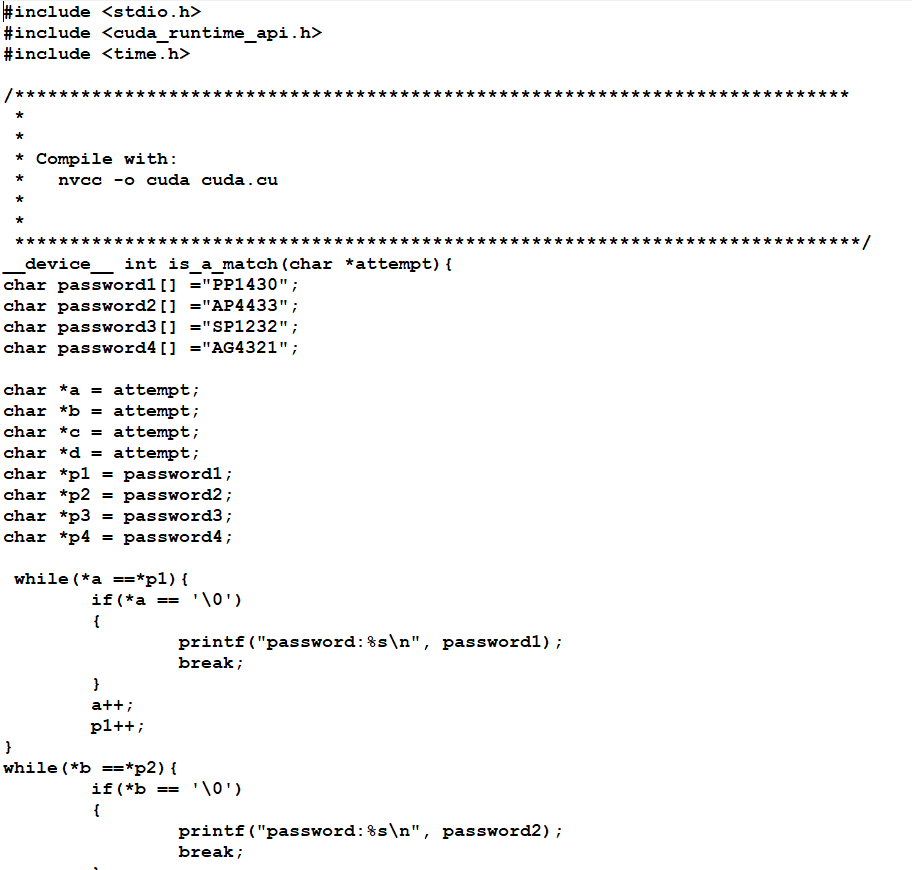
# **2.0 Cuda**

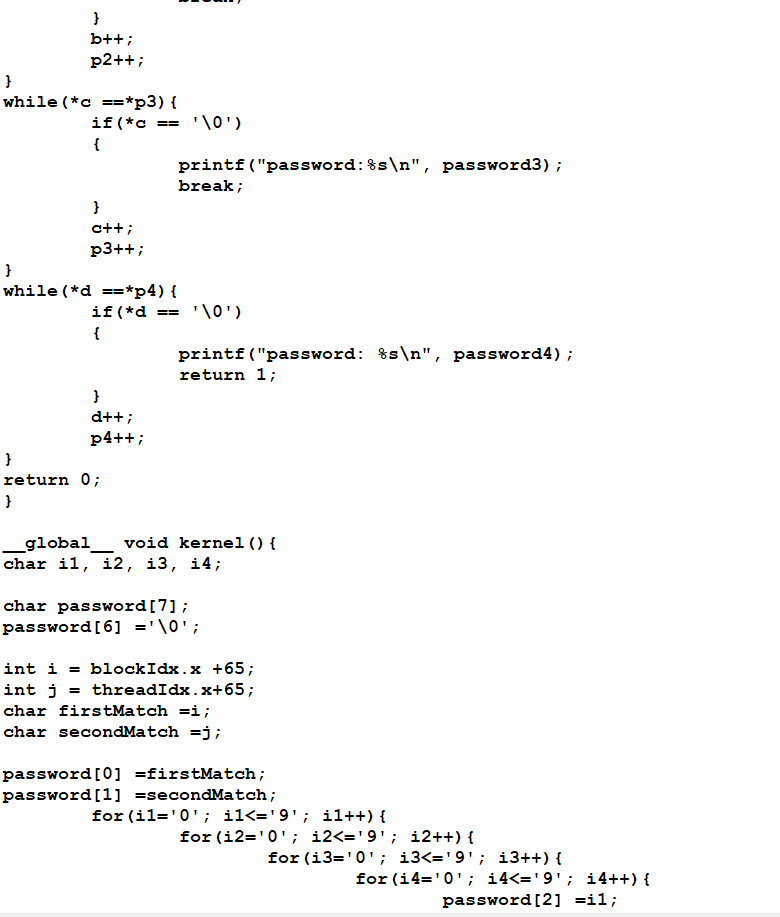
## **2.1 Password Cracking**

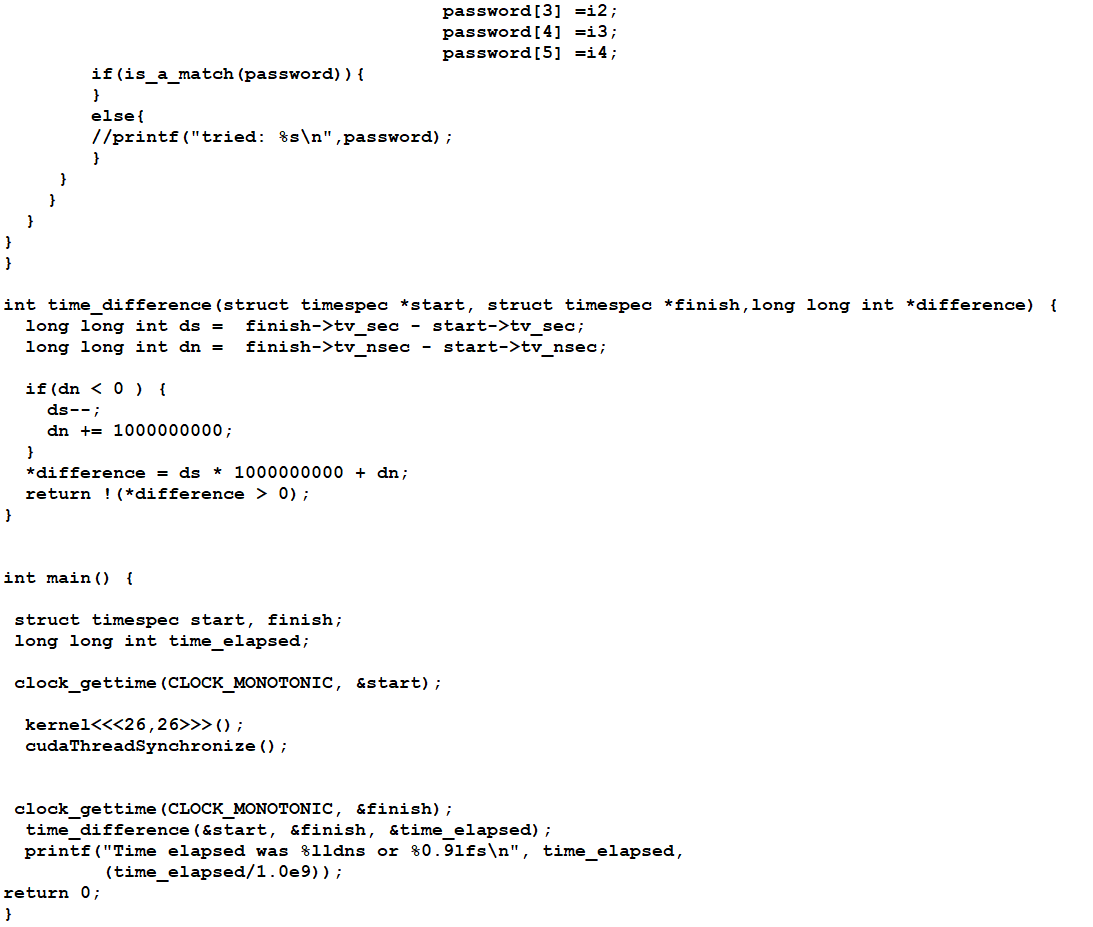
Compiling and running the program.



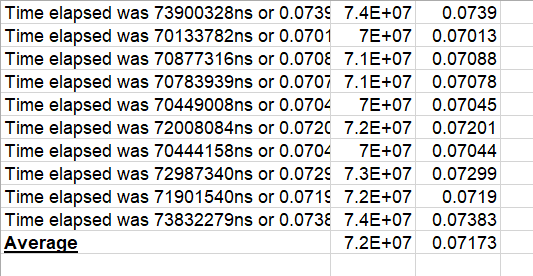
**Source Code:**





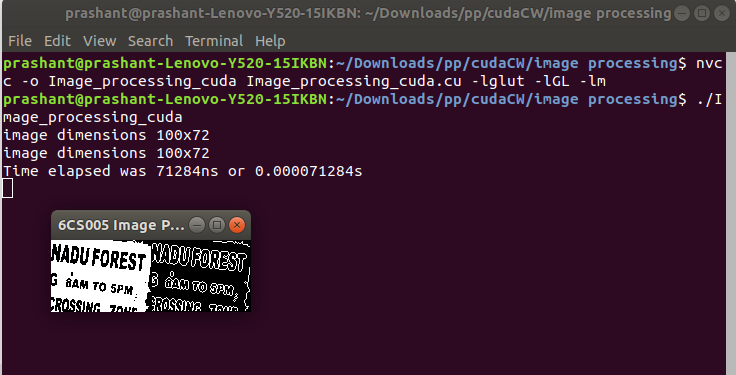


Time taken to run the program

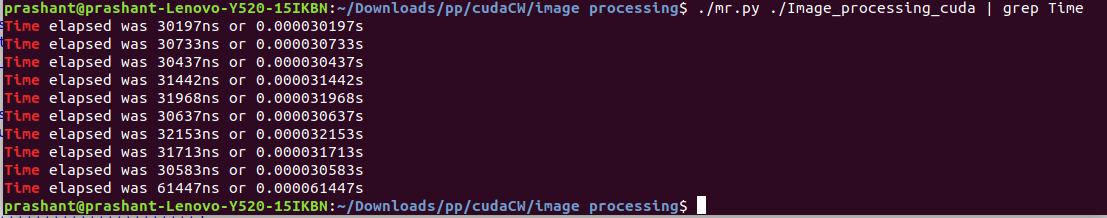


## **2.2 Image Processing**

Compile and run the program

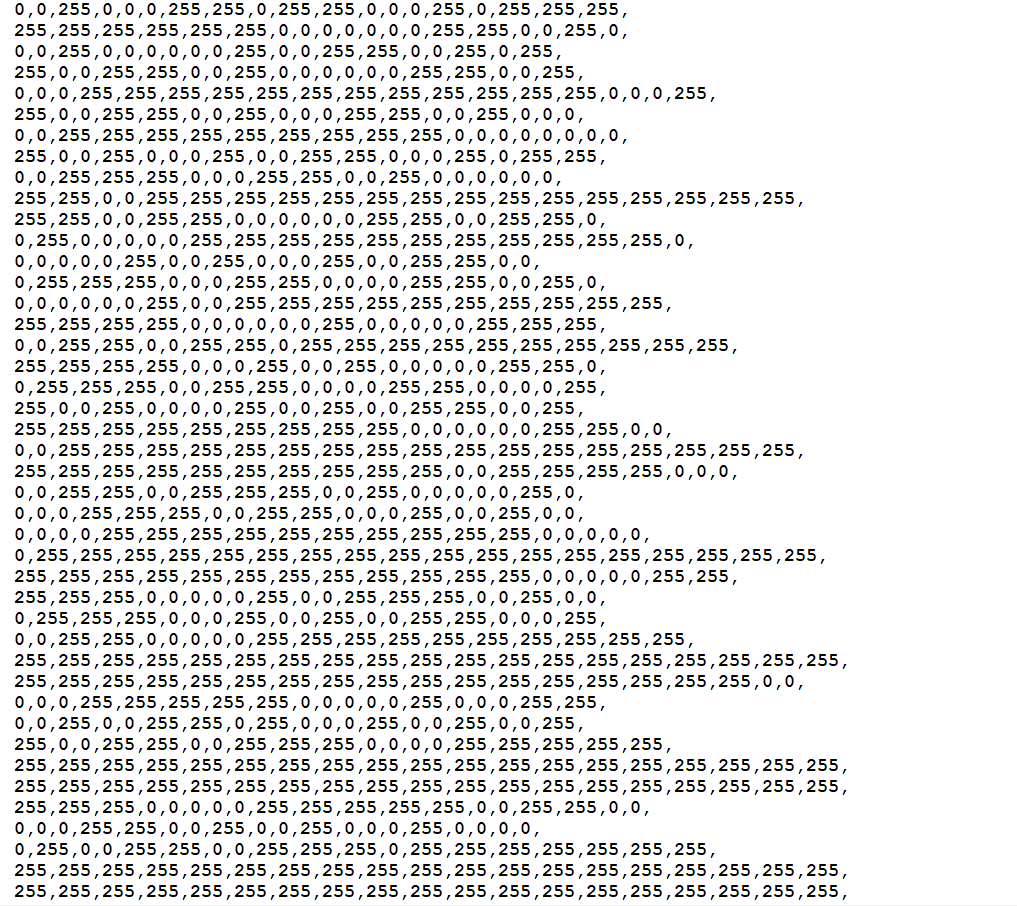


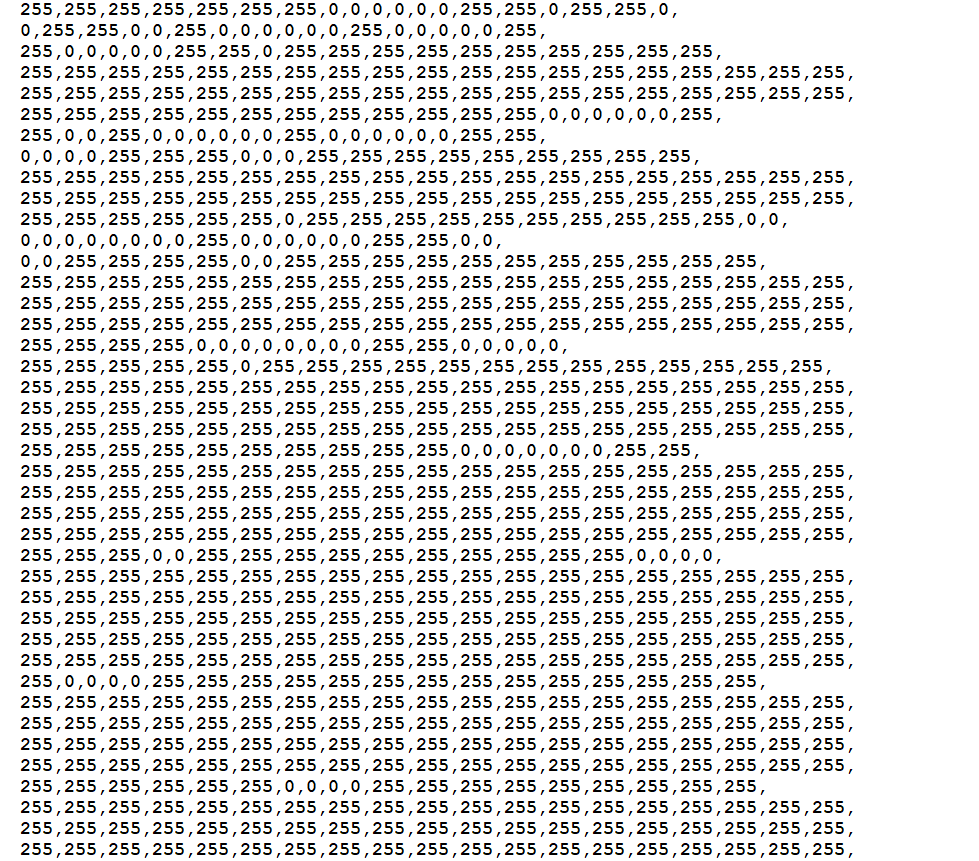
Run the program 10 times

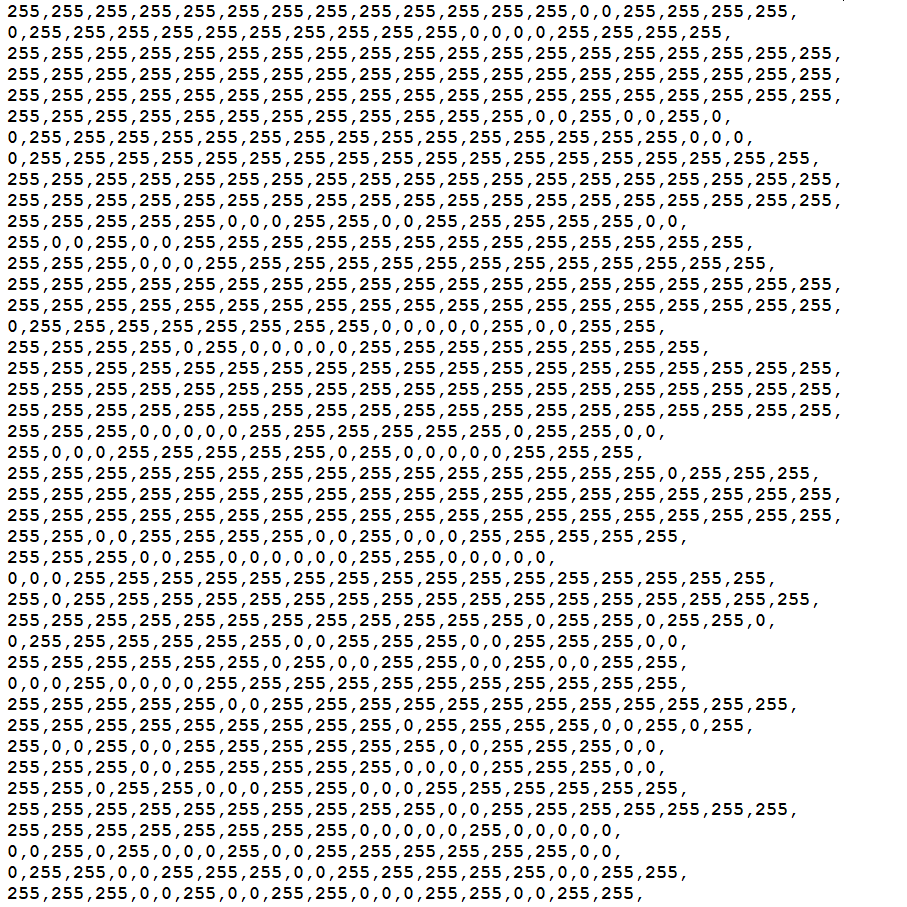


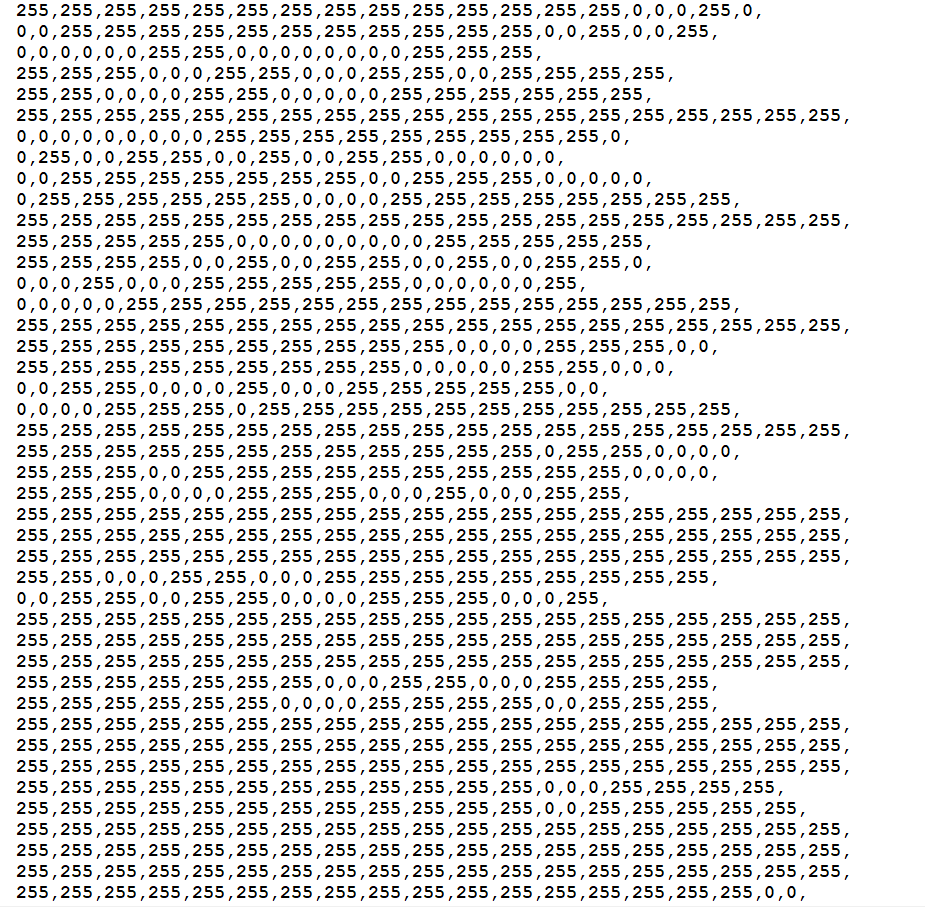
**Source Code**

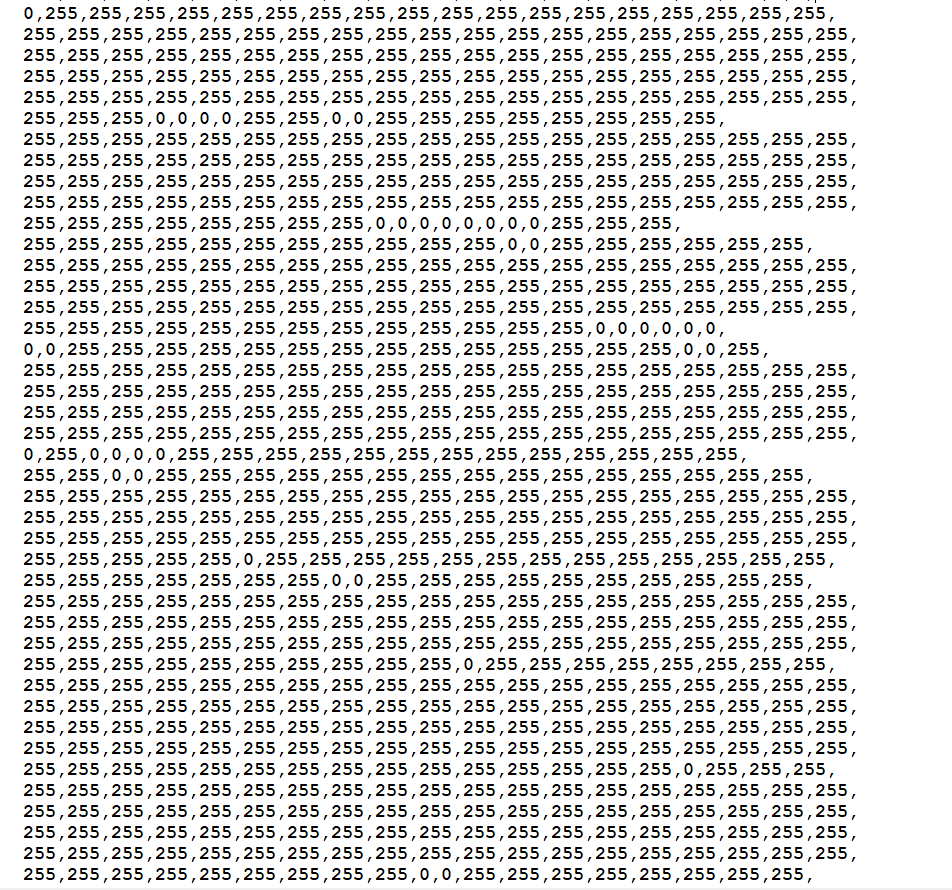


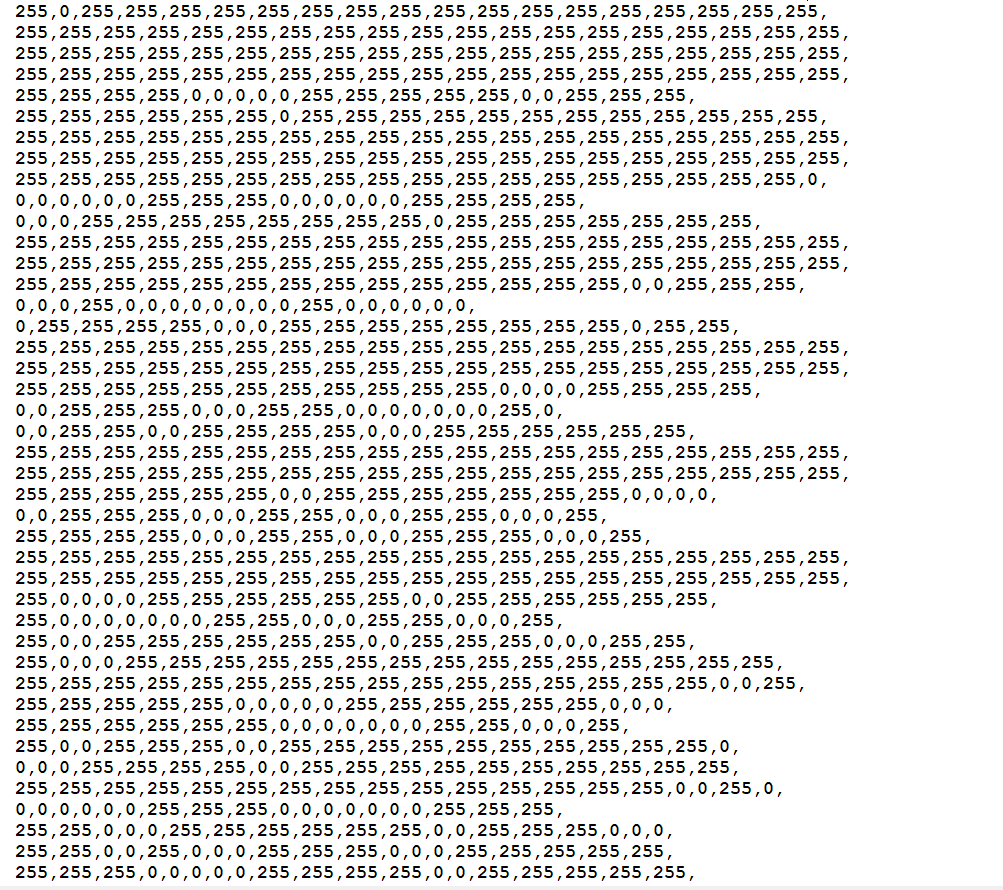


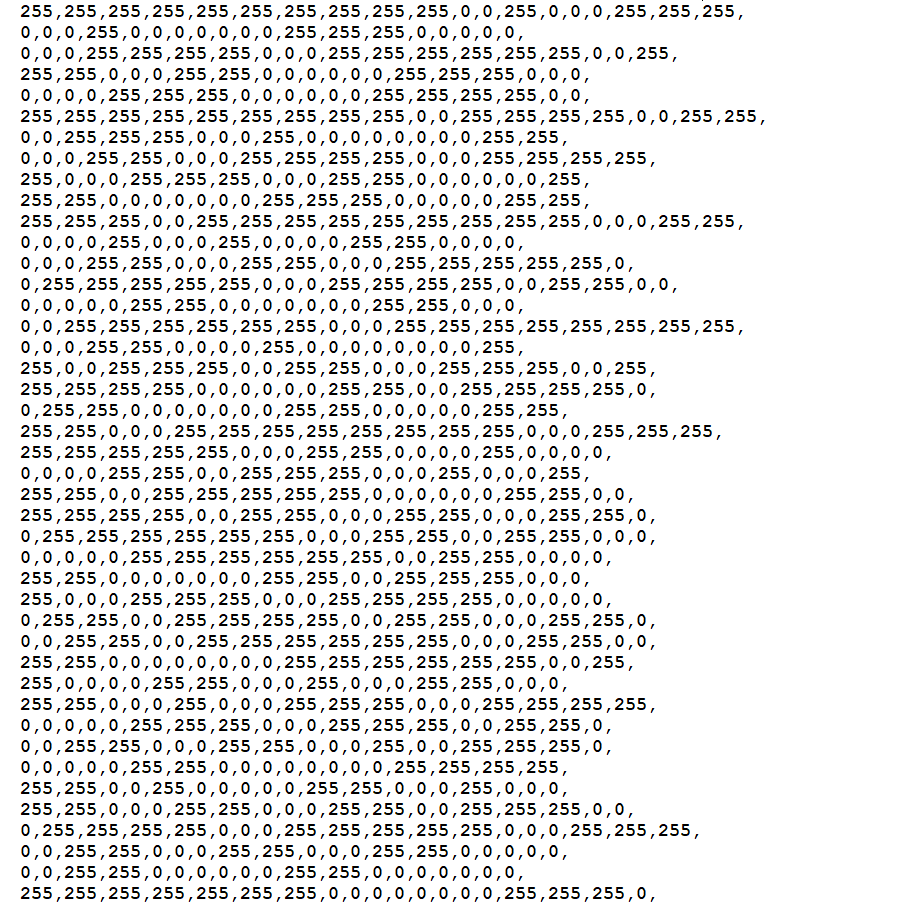


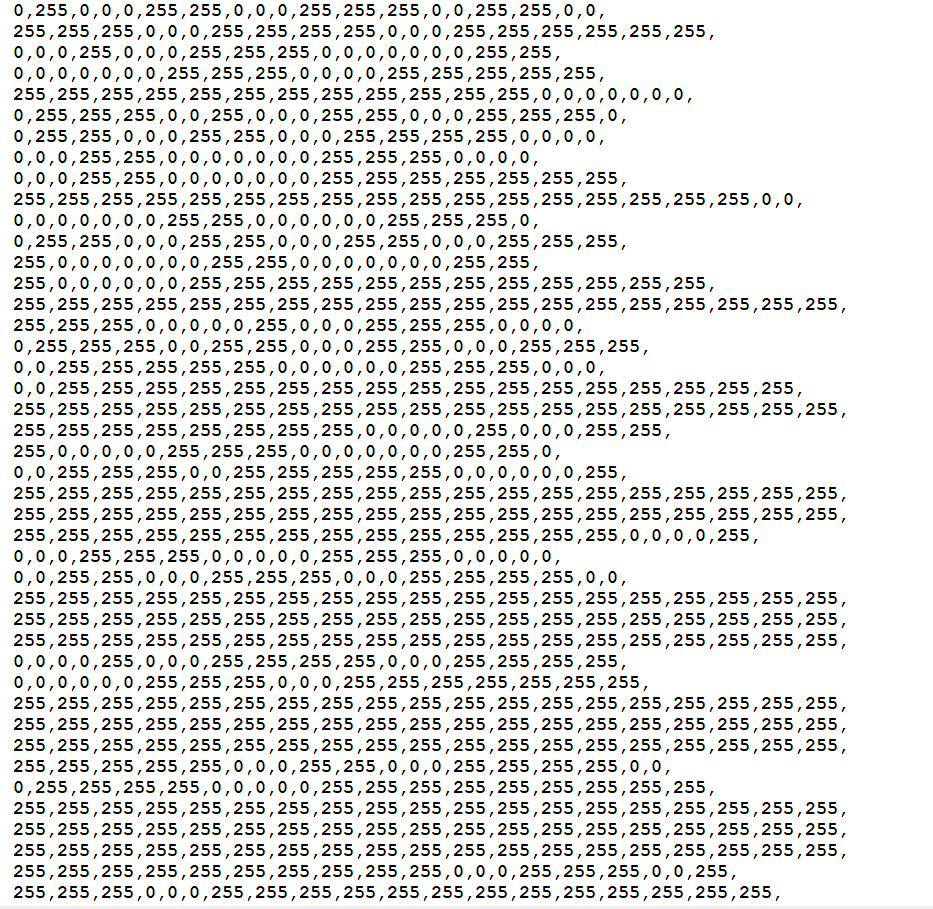


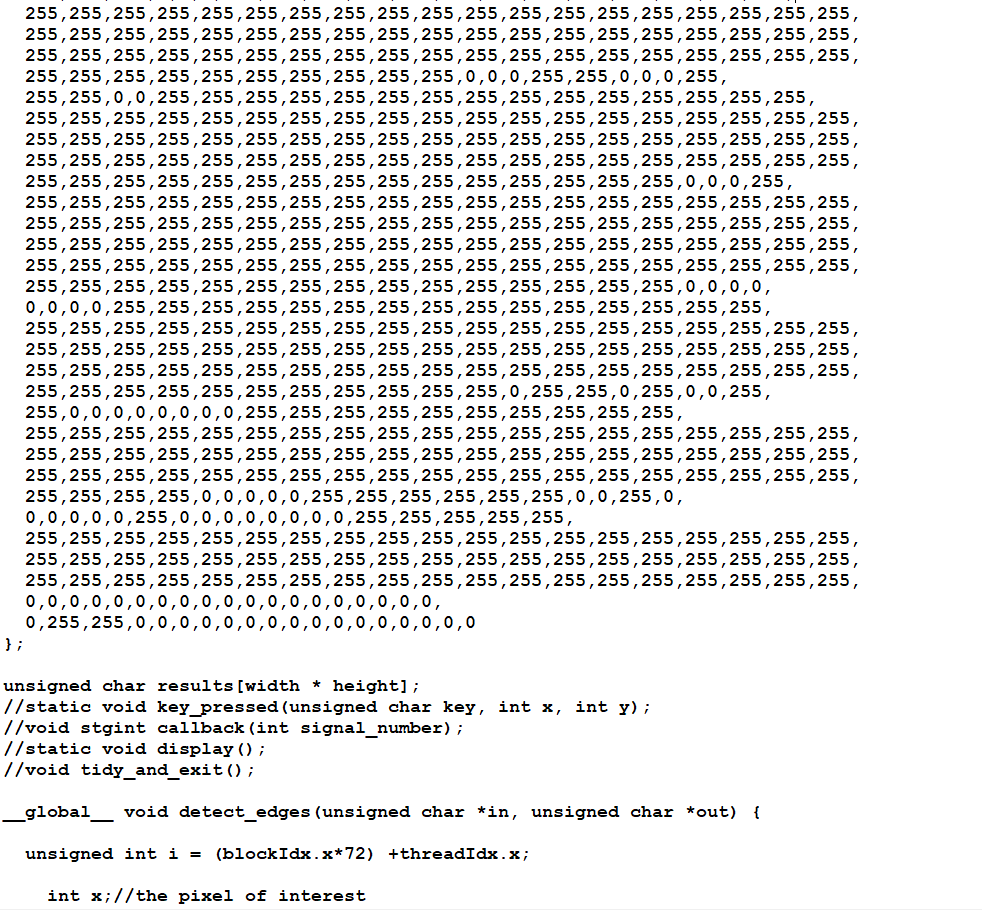


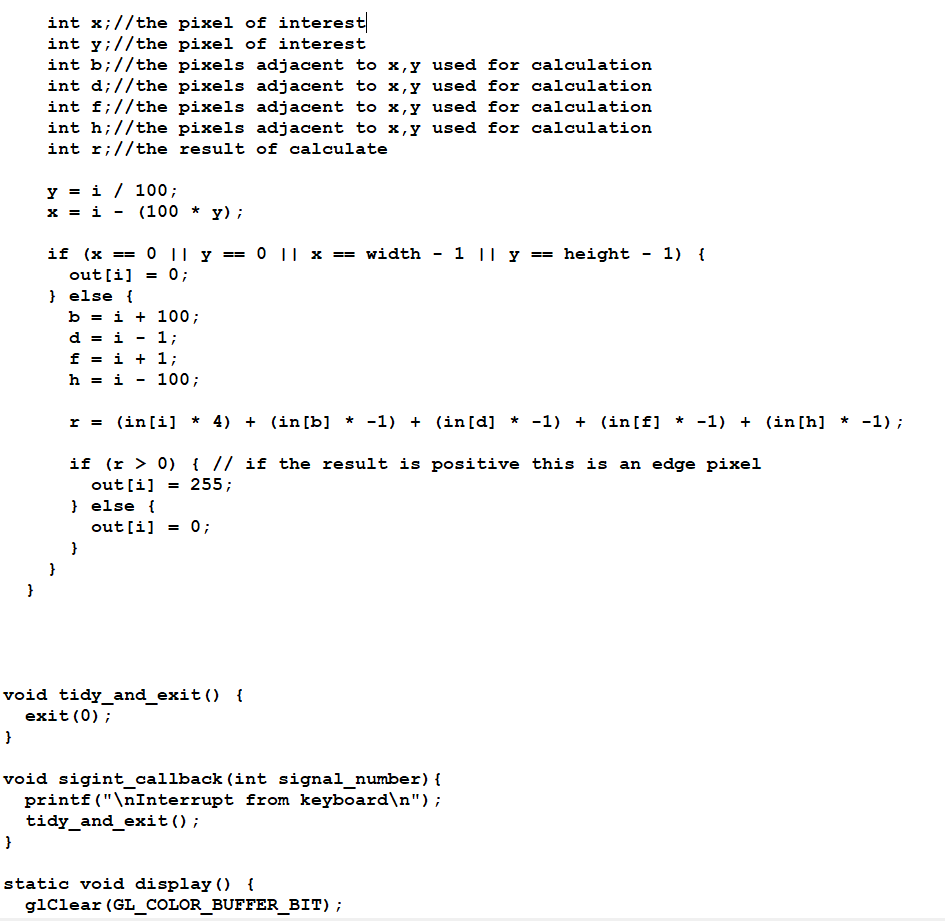




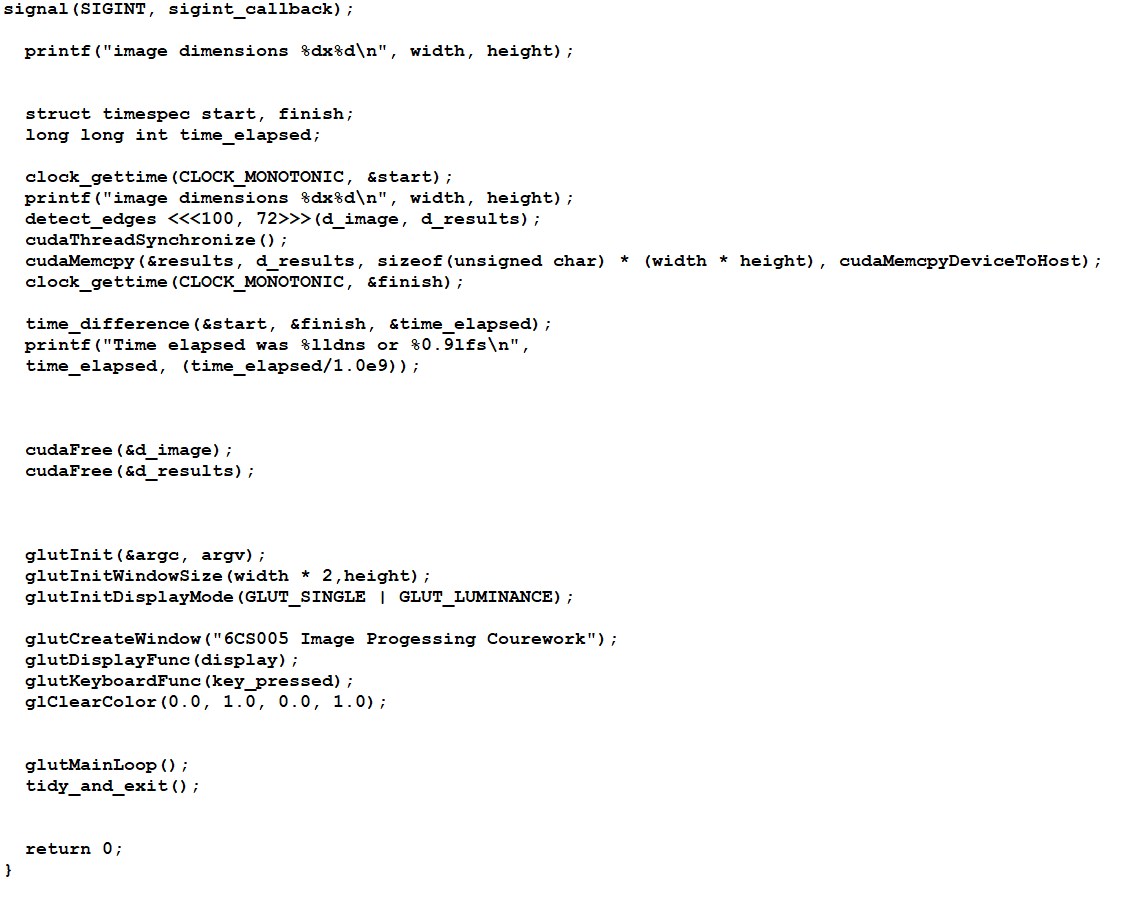






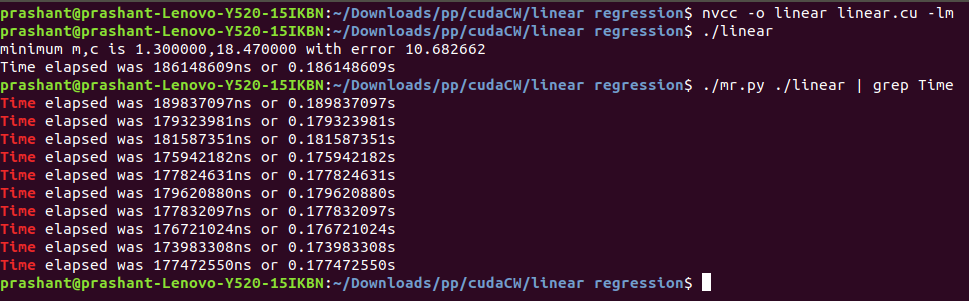




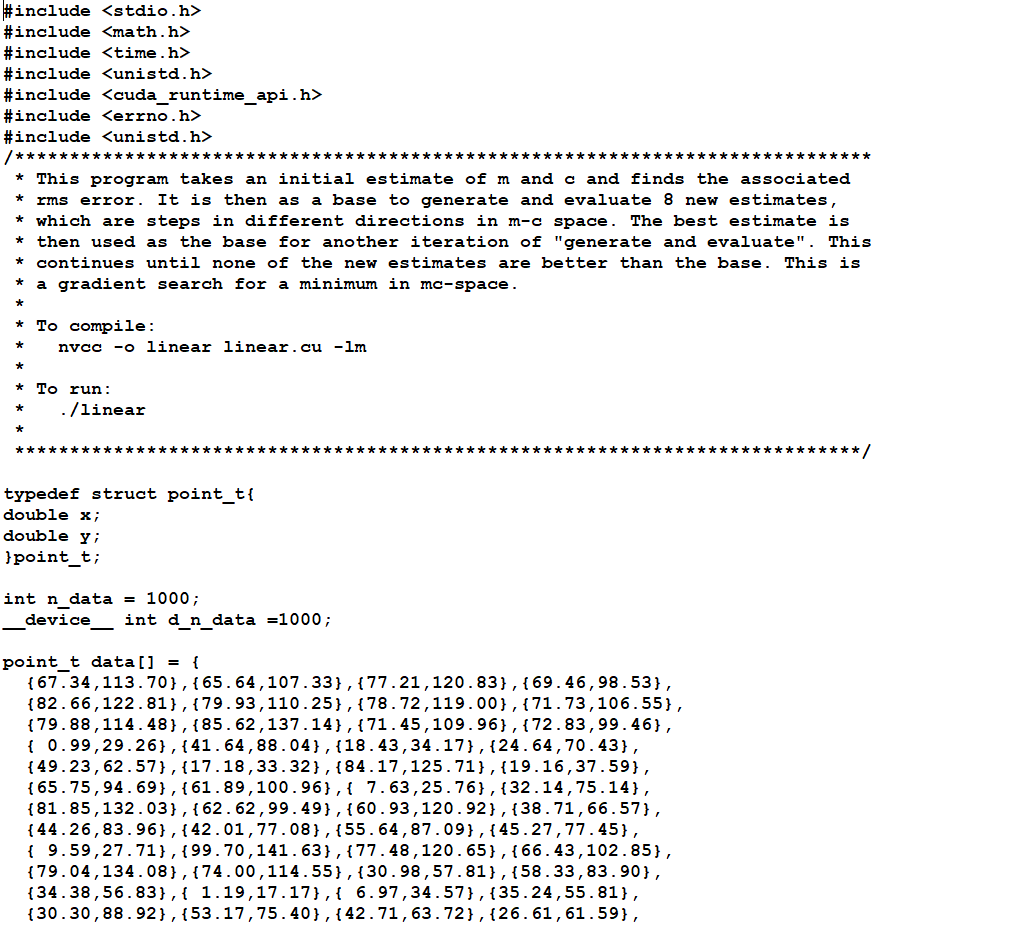


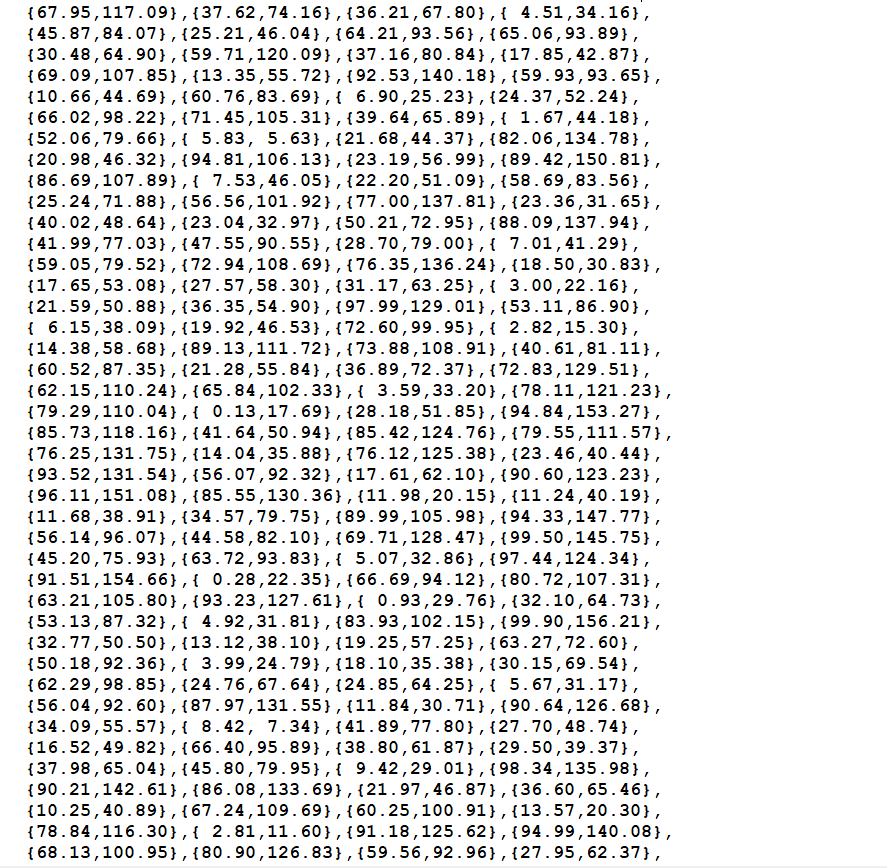
## **2.3 Linear Regression**

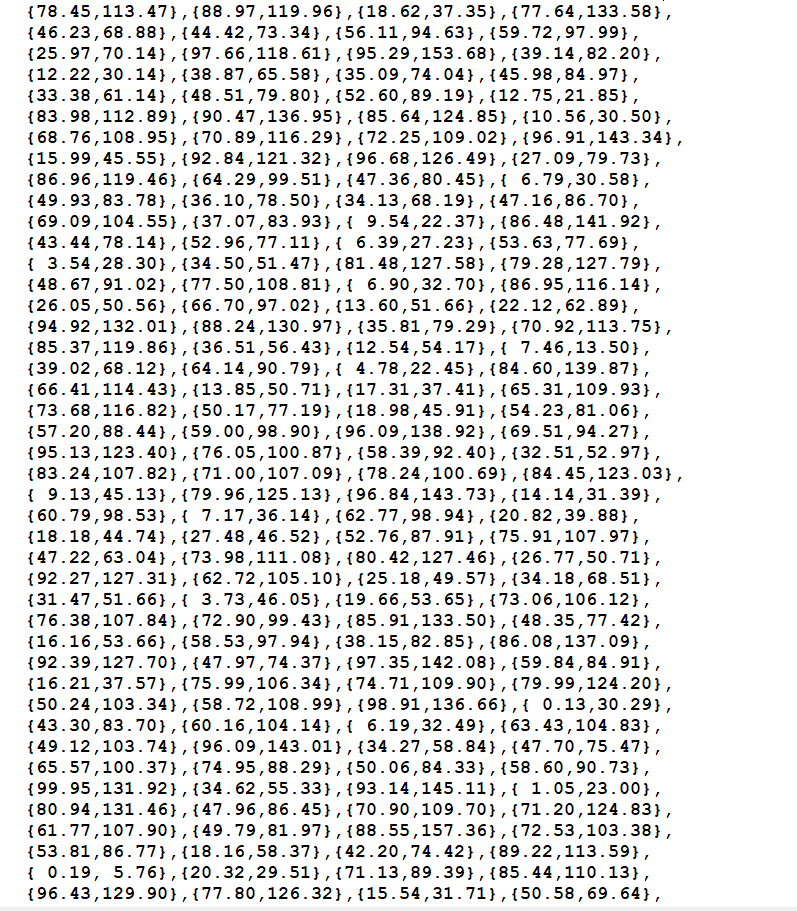
Compile and run the program

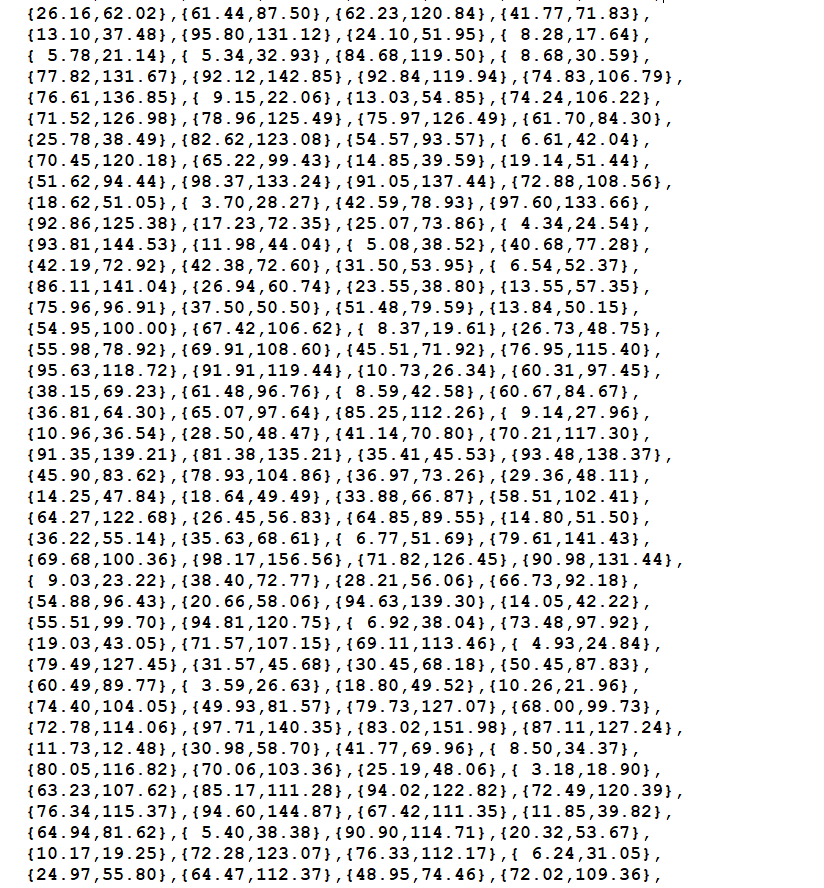


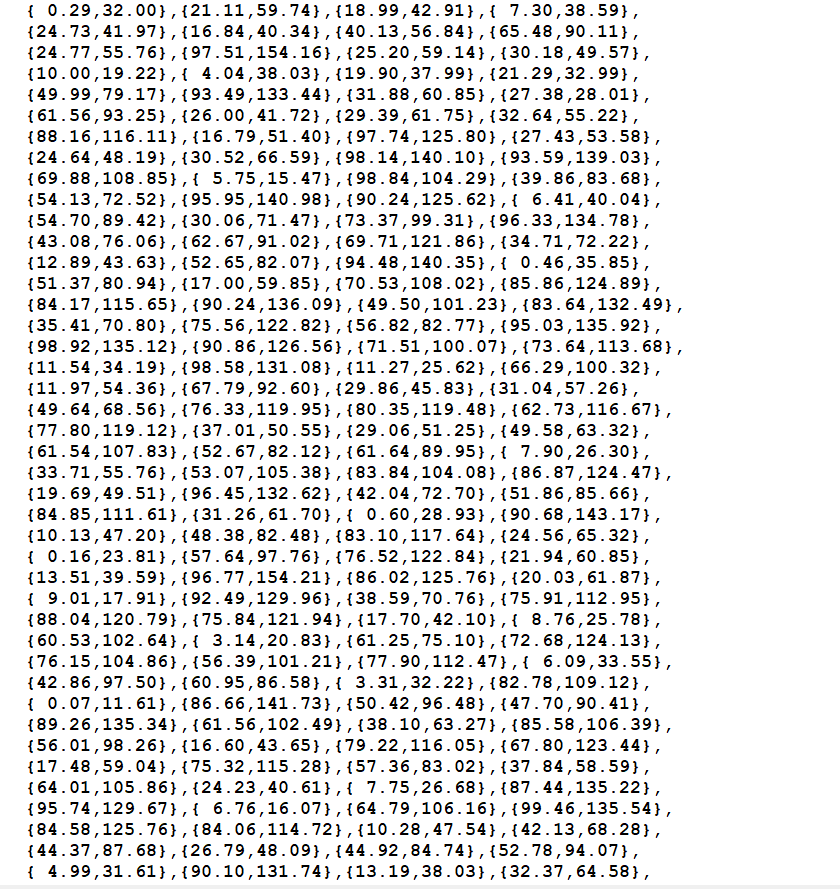
Source Code

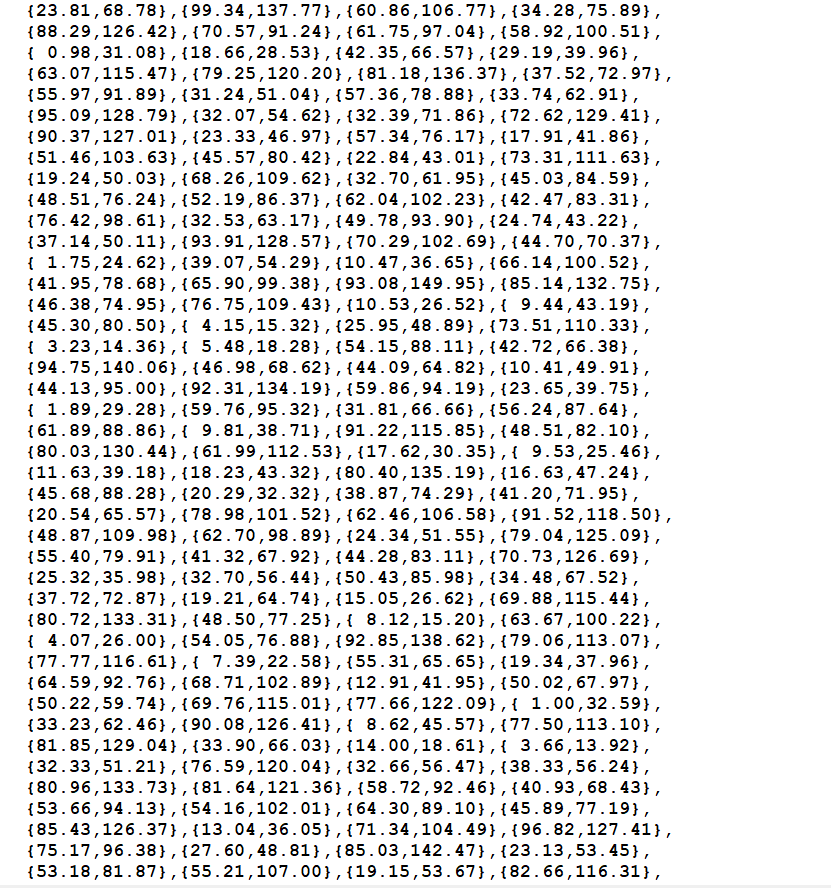


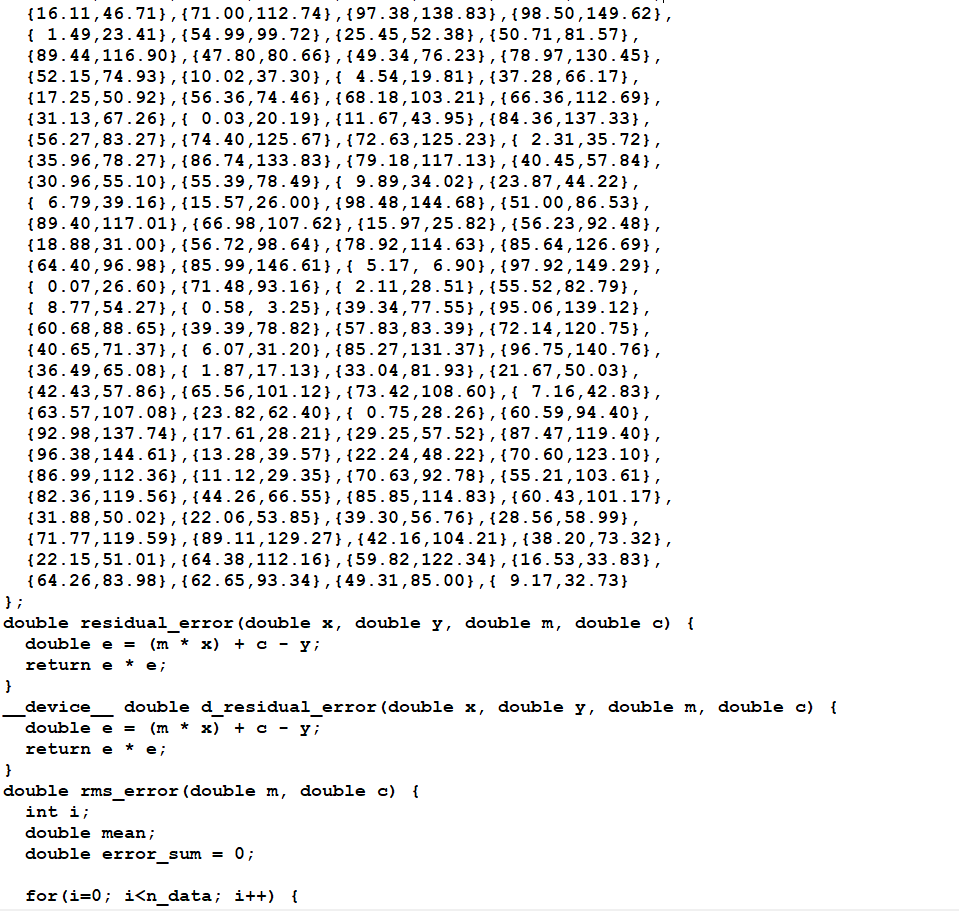


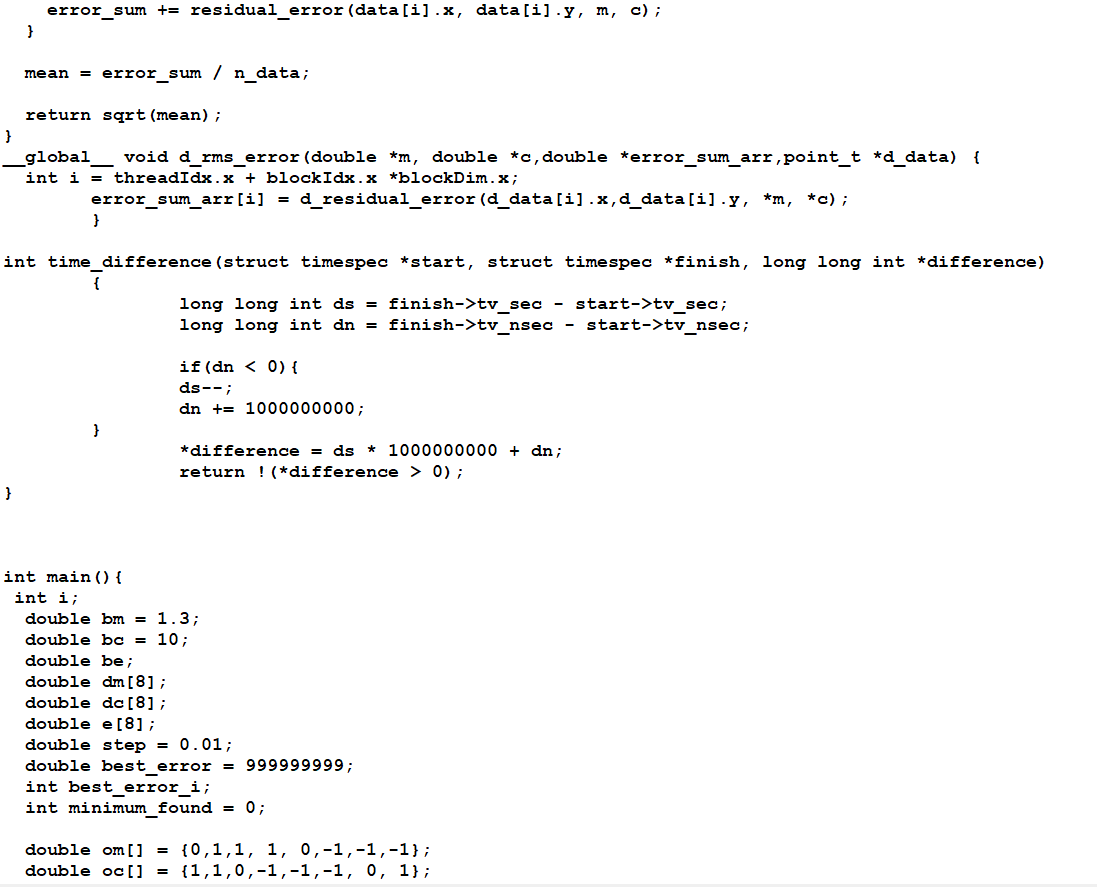


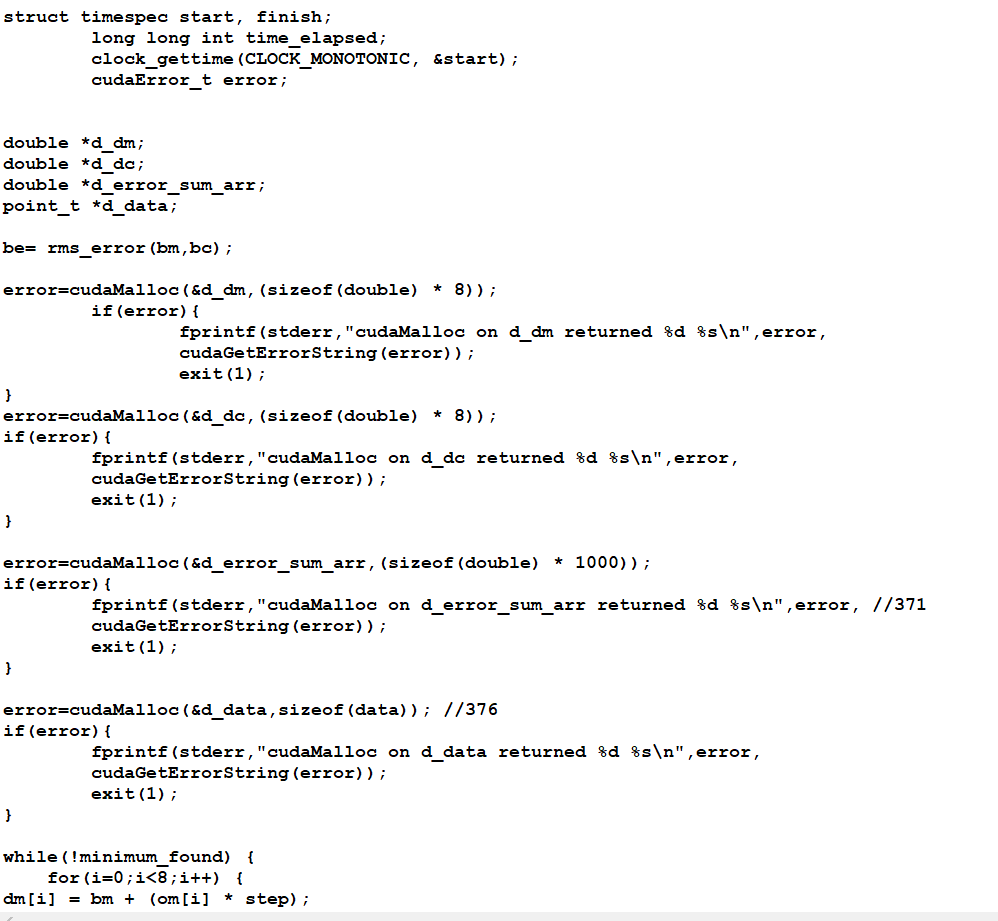


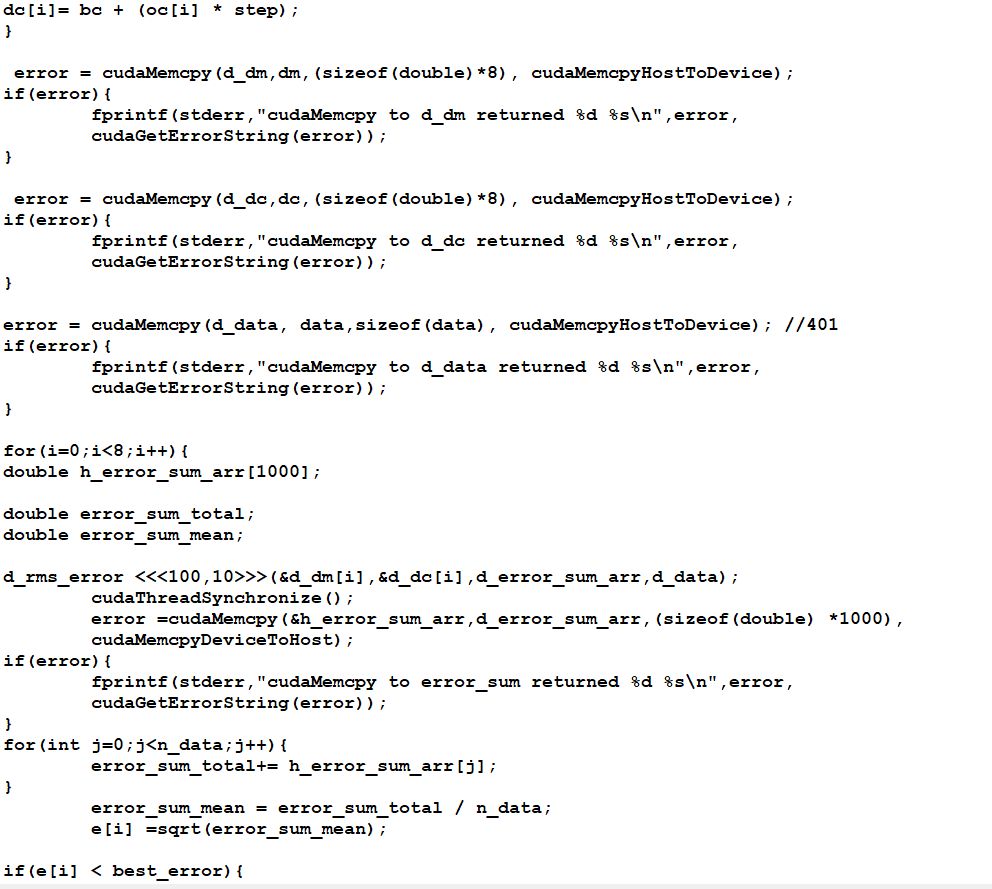


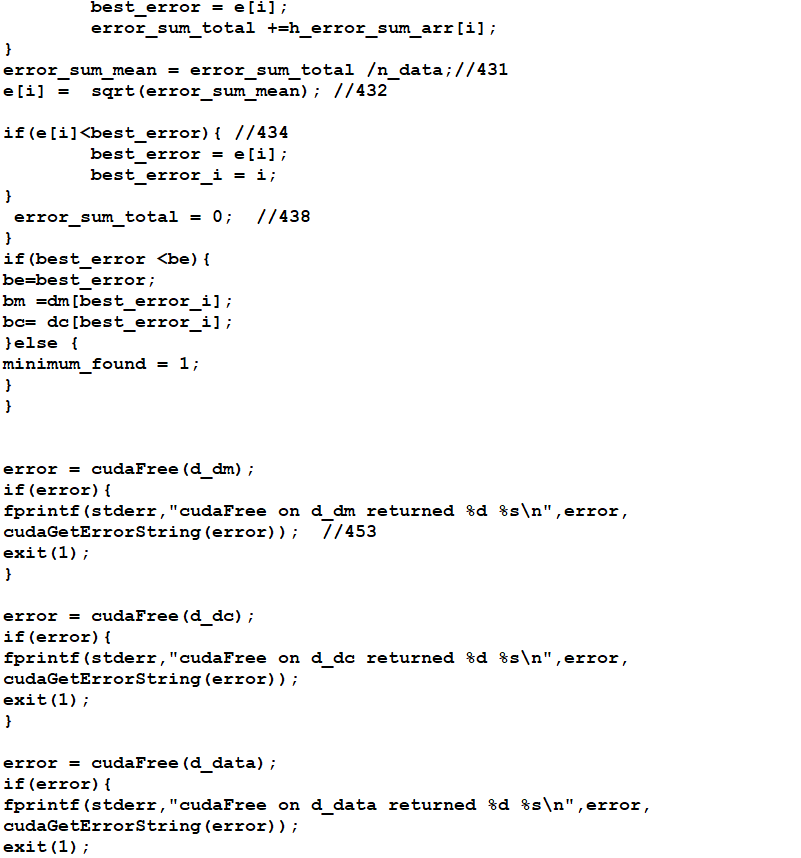


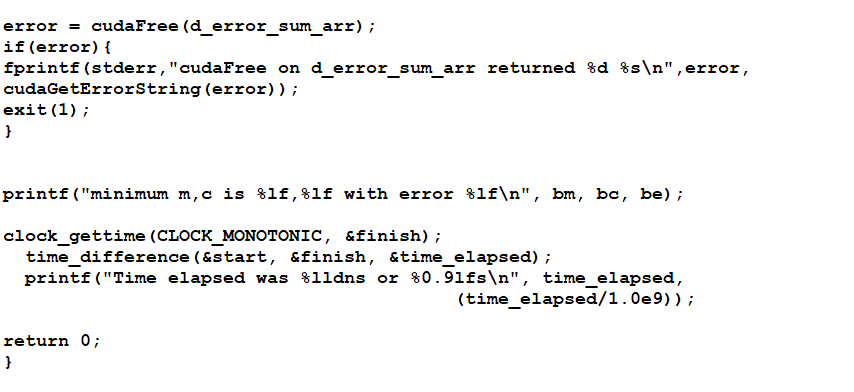








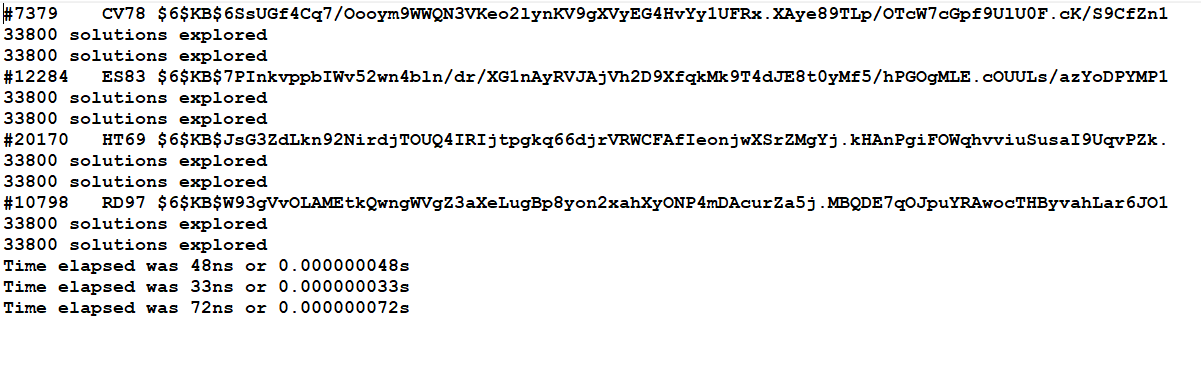




# **3.0 Message Passing Interface (MPI)**

## **3.1 Password Cracking**

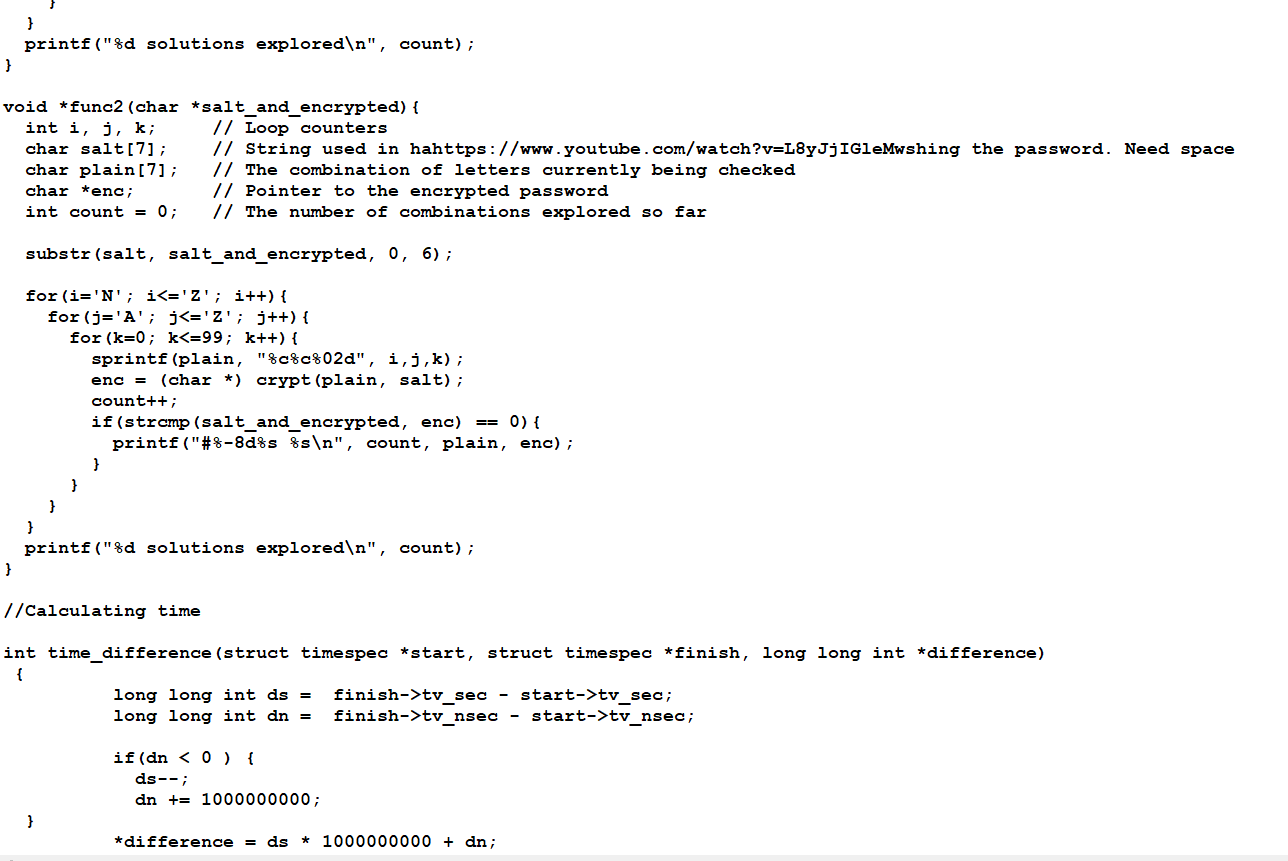
Running the program



**Source Code**

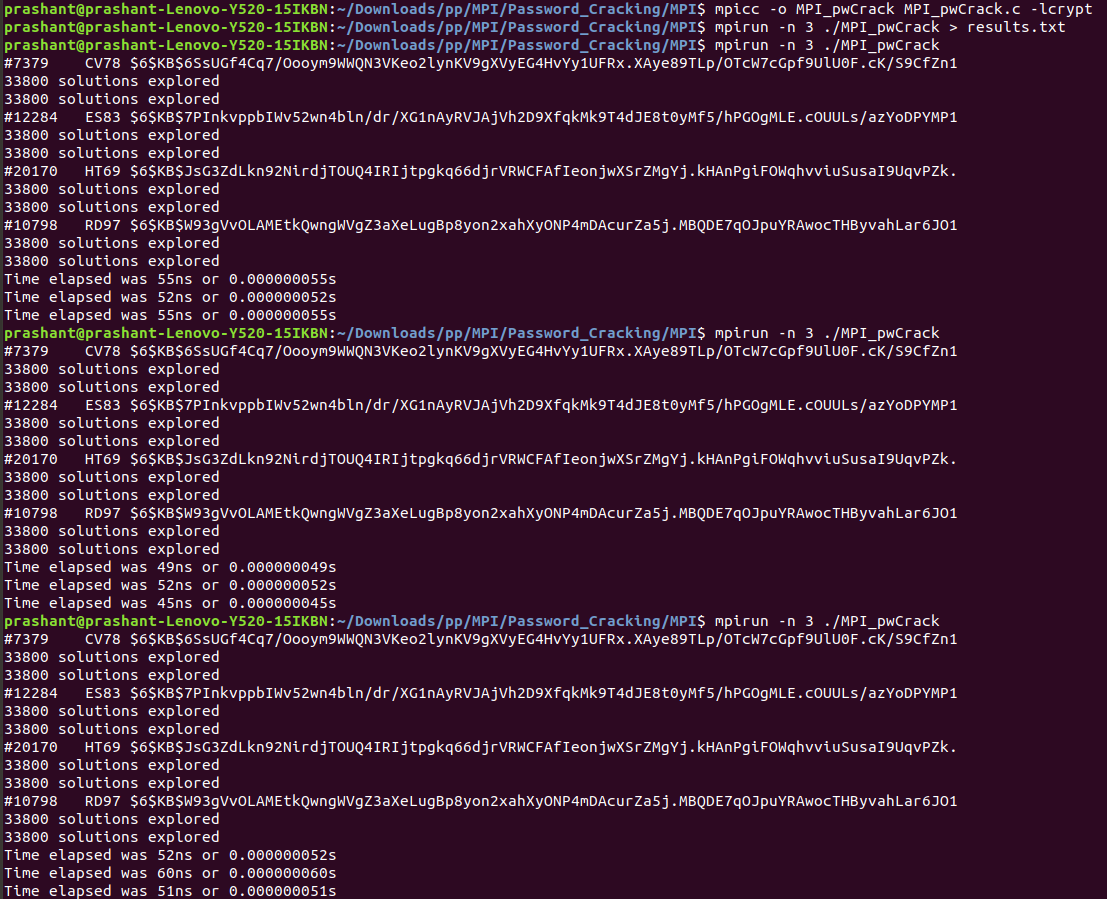
****

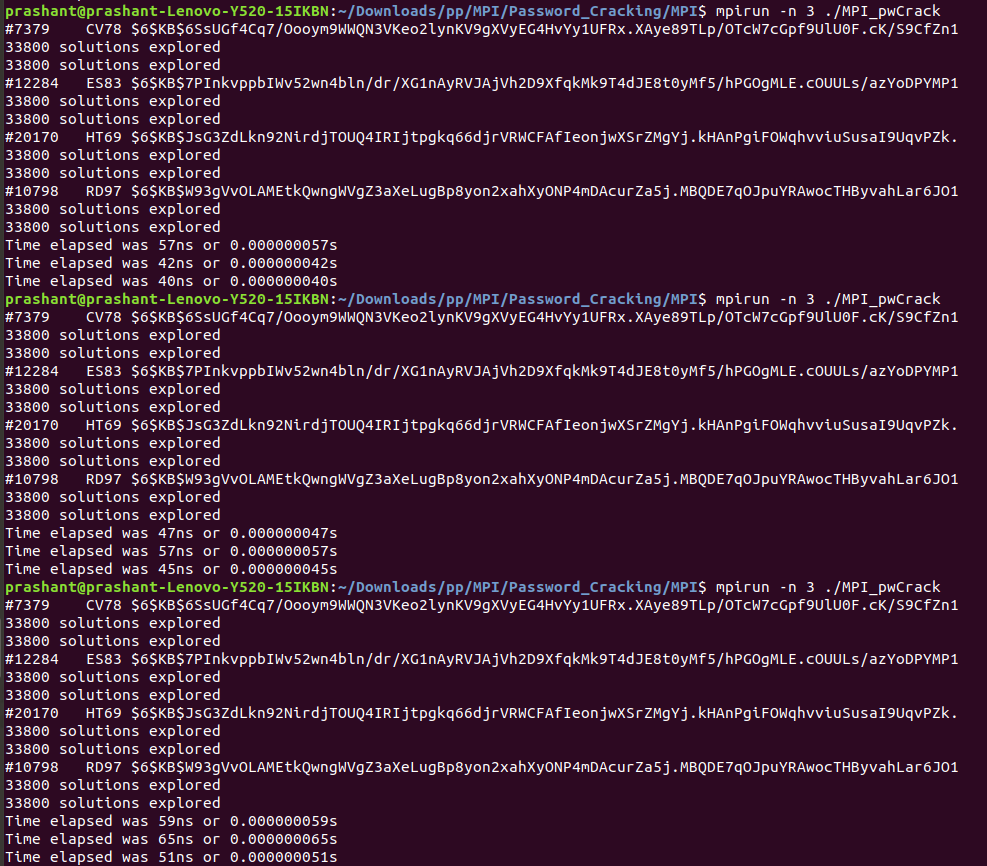
****

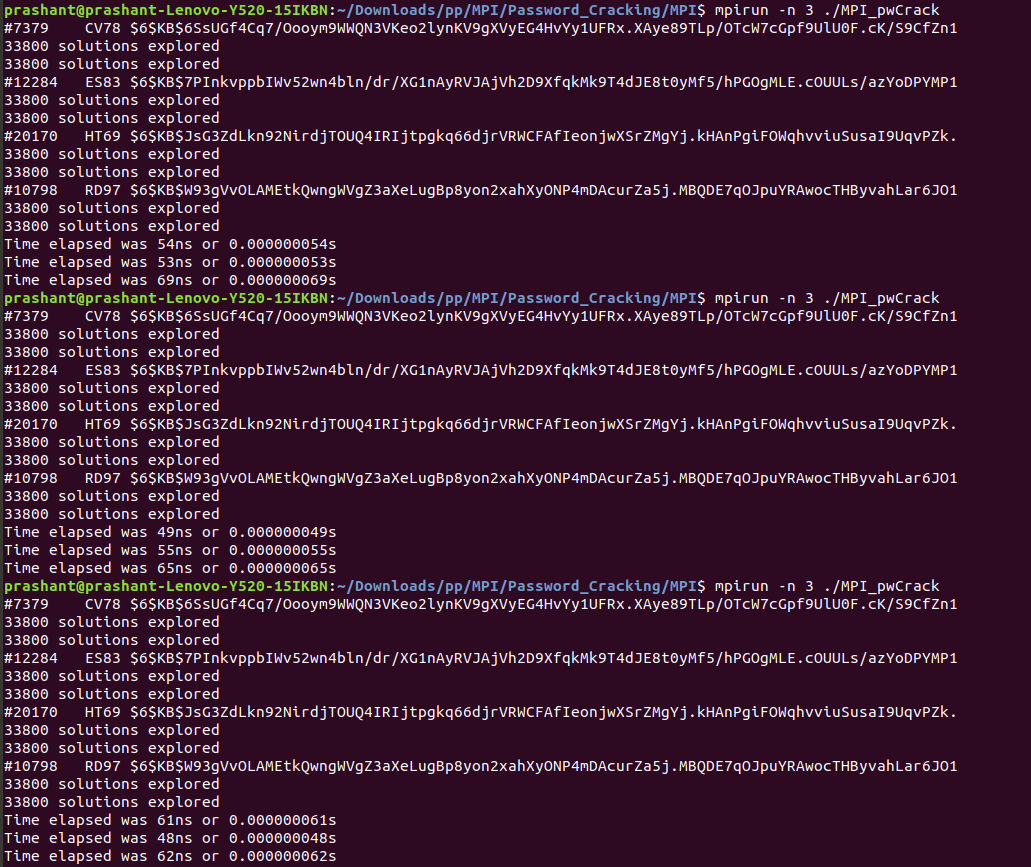
****

****

Compiling and running the program 10 times.

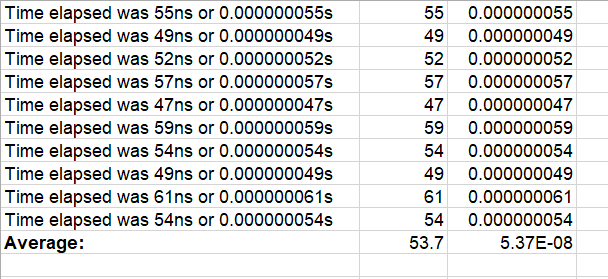








Calculating the average time.

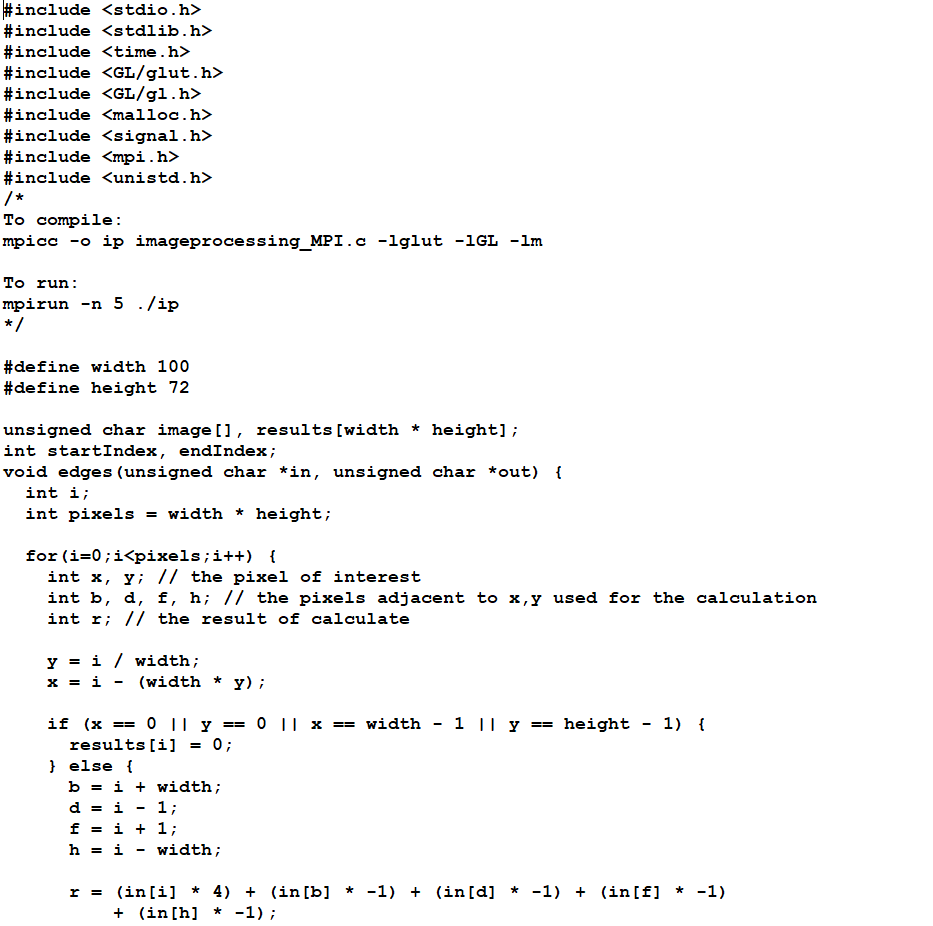


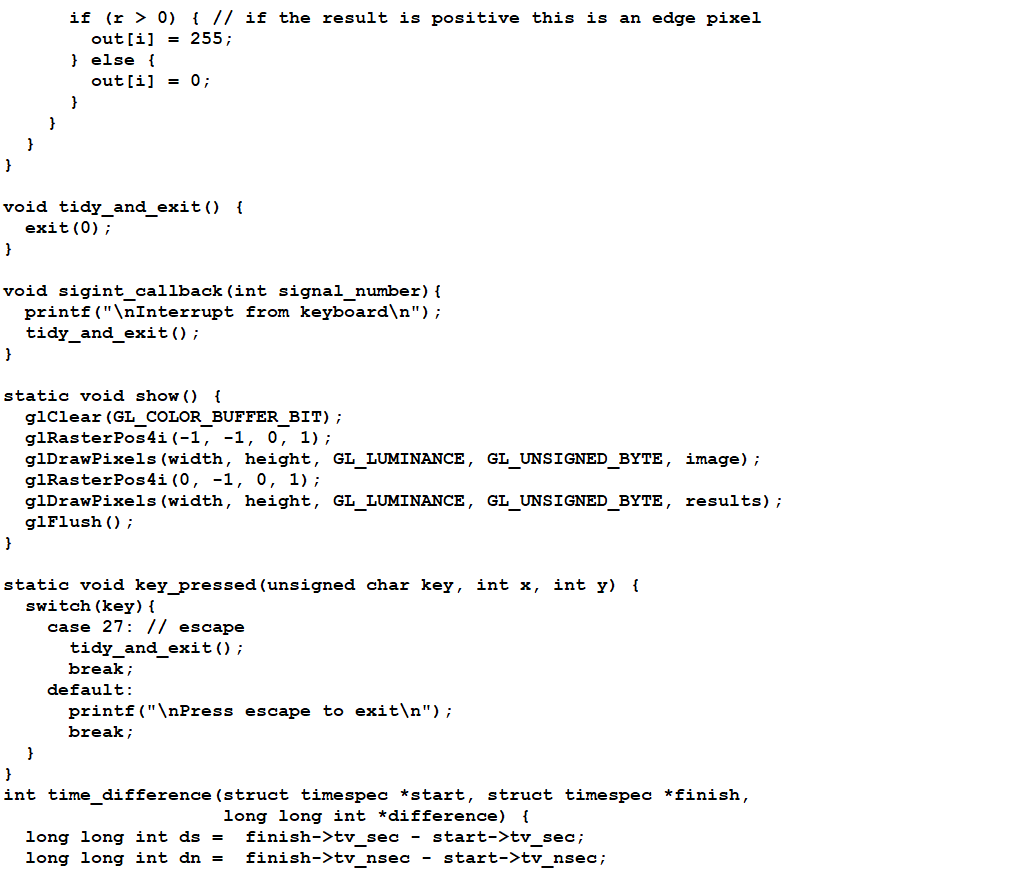
## **3.2 Image Processing**

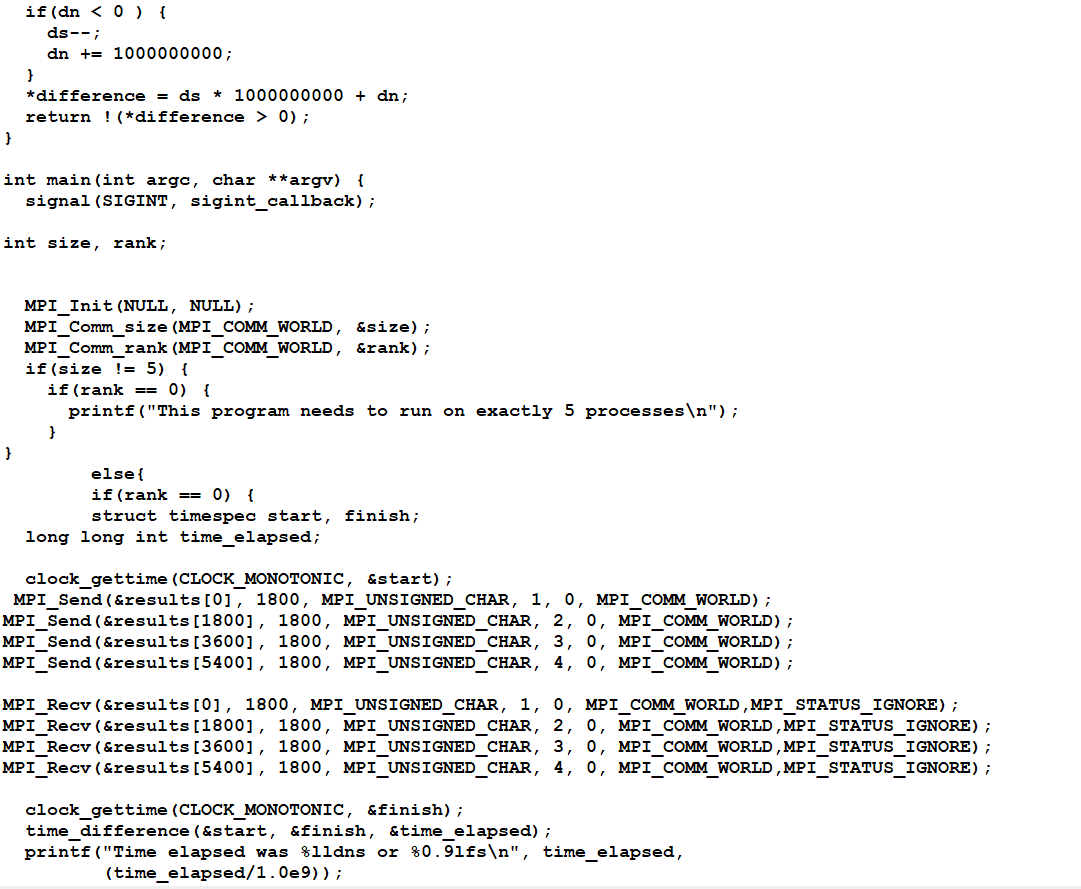
Compiling and running the program

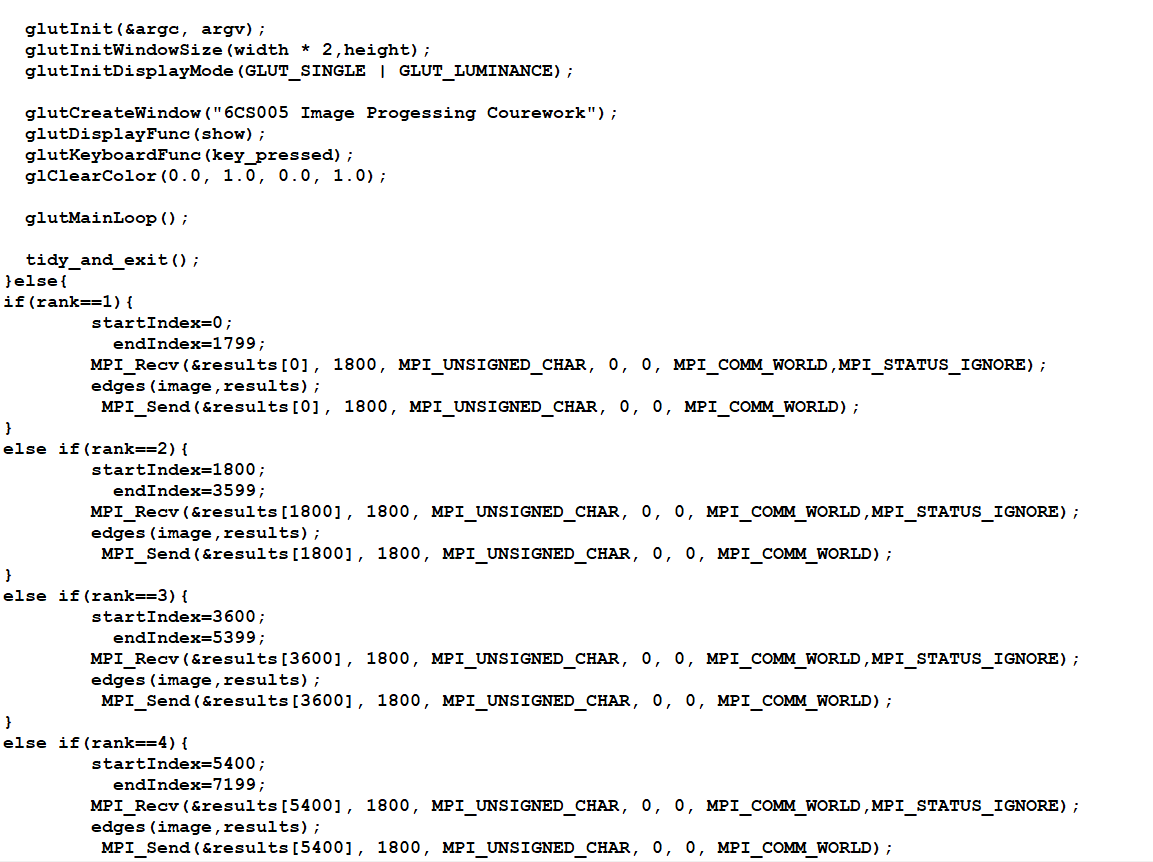


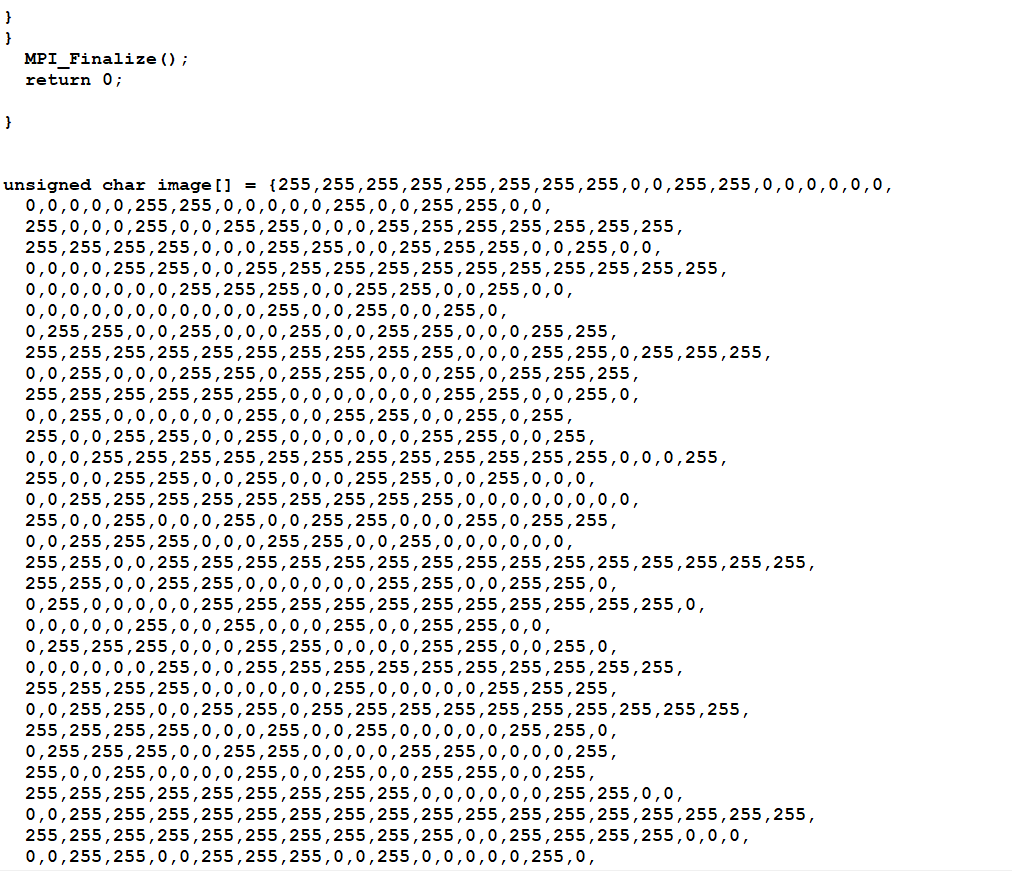
**Source code**

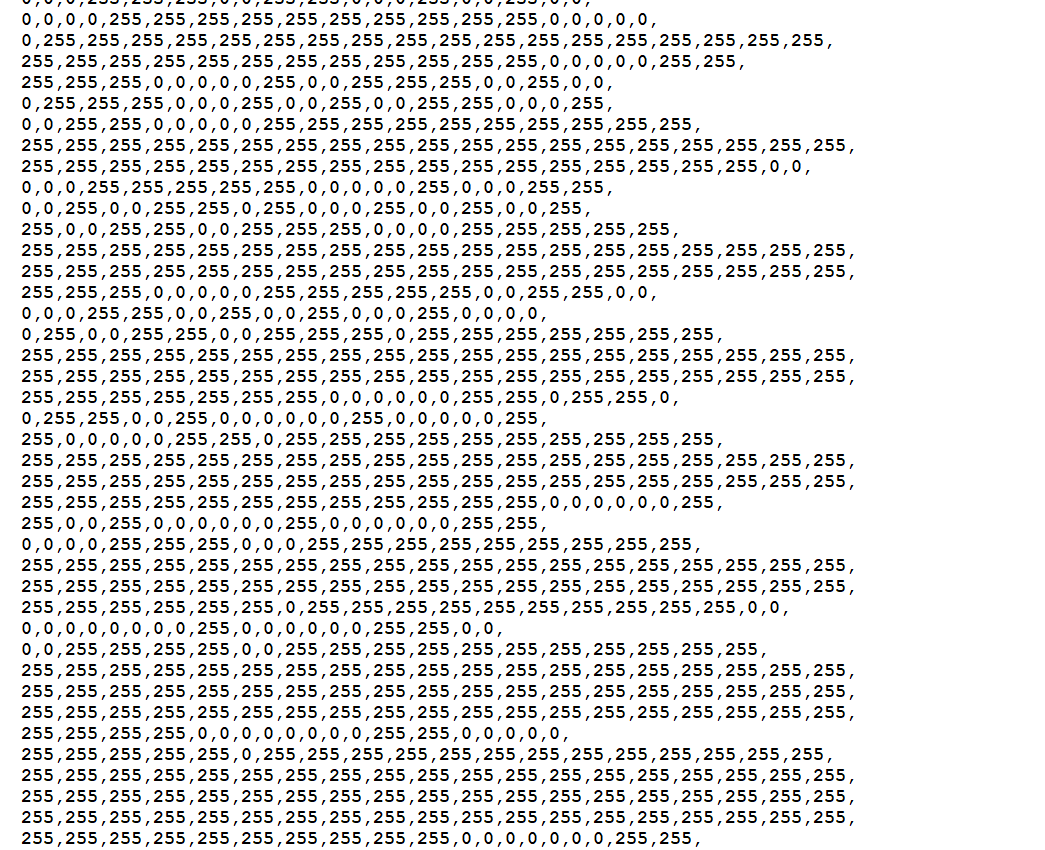
****

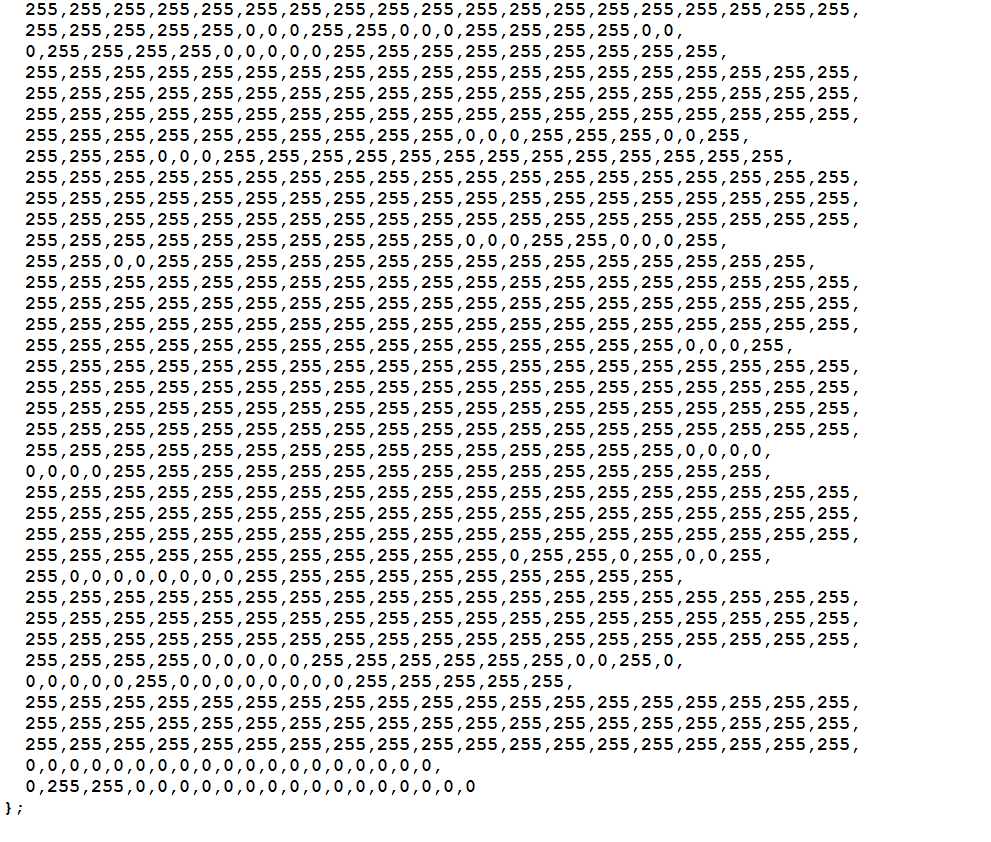
****

****

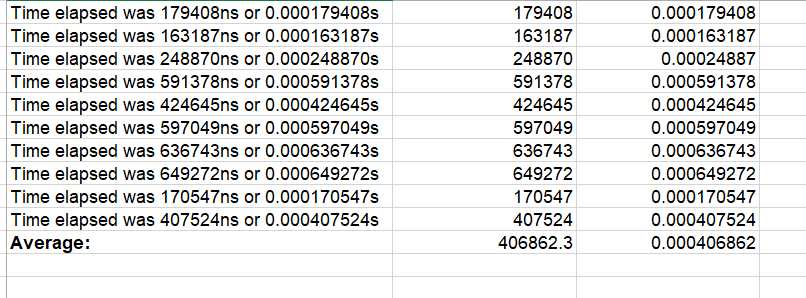
****

****

****

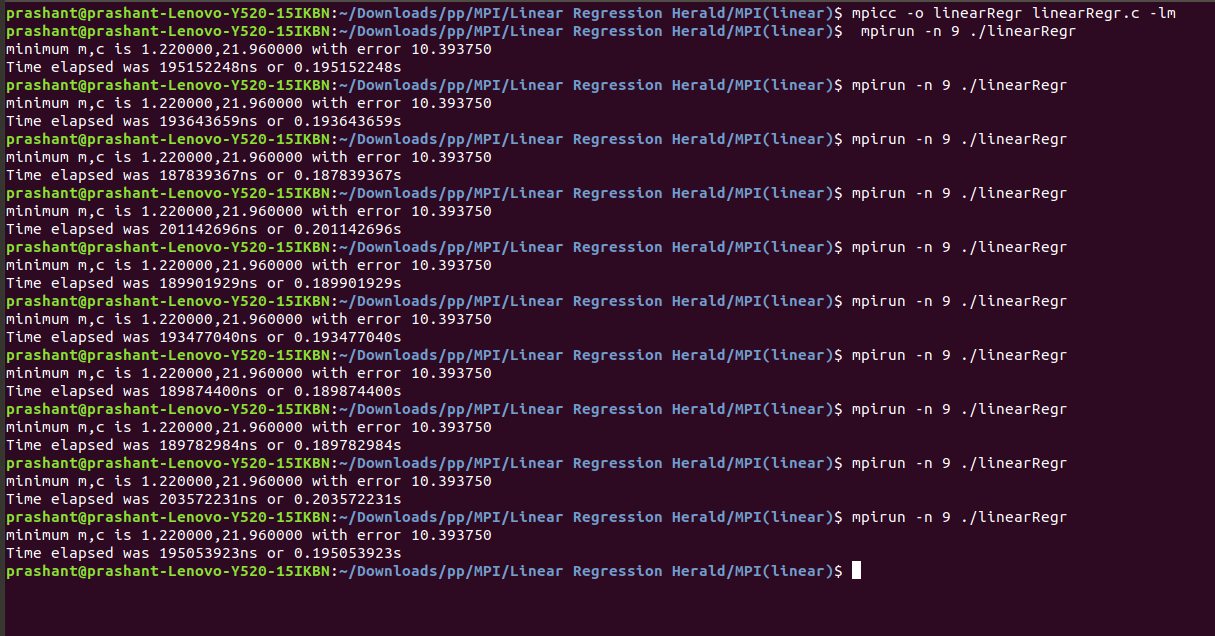
****

**Calculating average time**

****

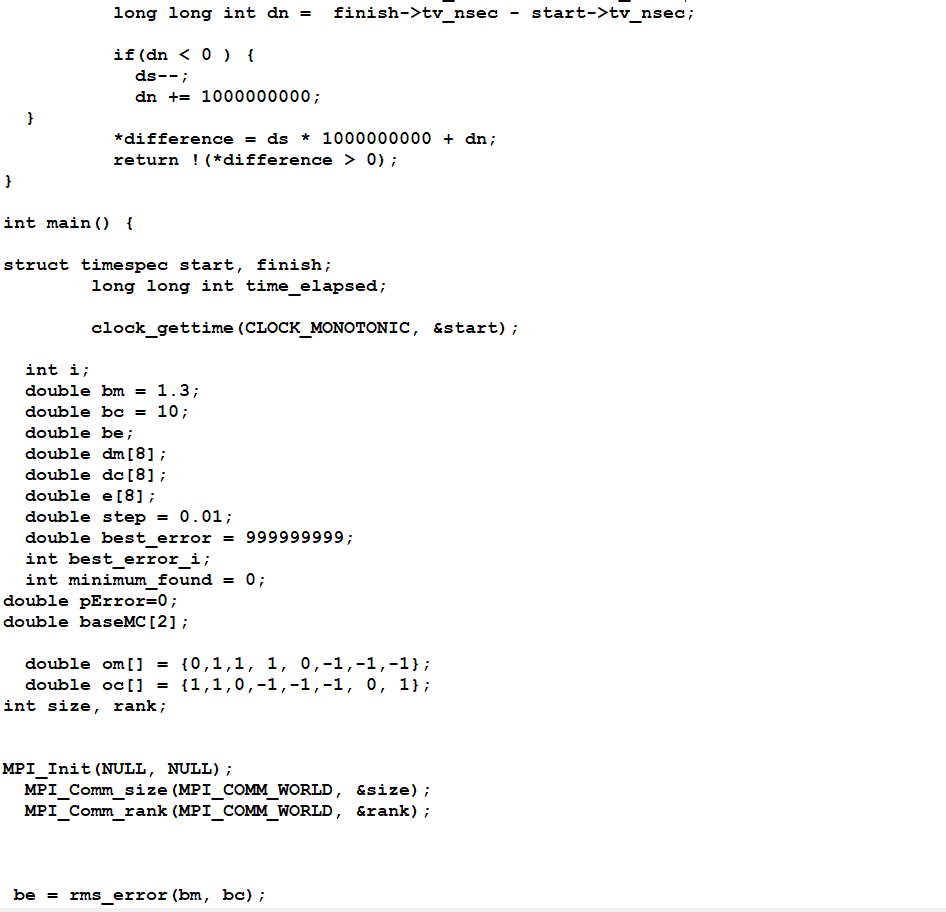
## **3.3 Linear Regression**

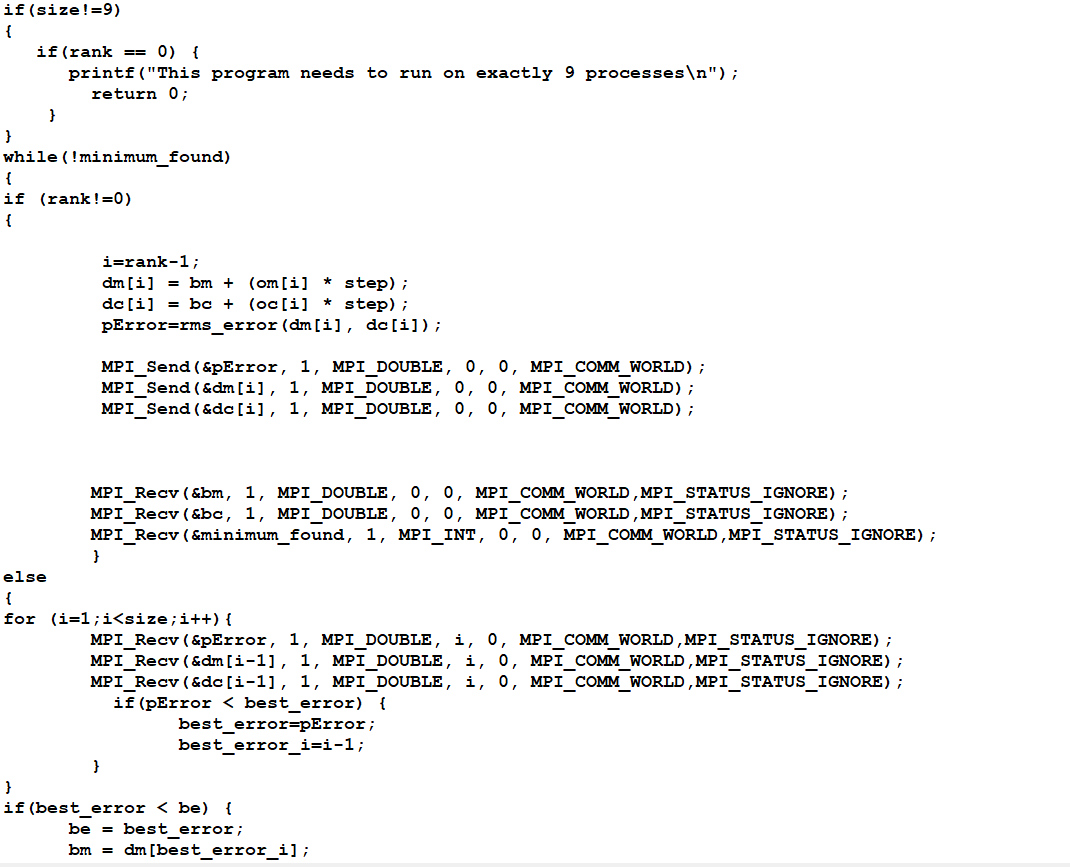
Compiling and running the program 10 times

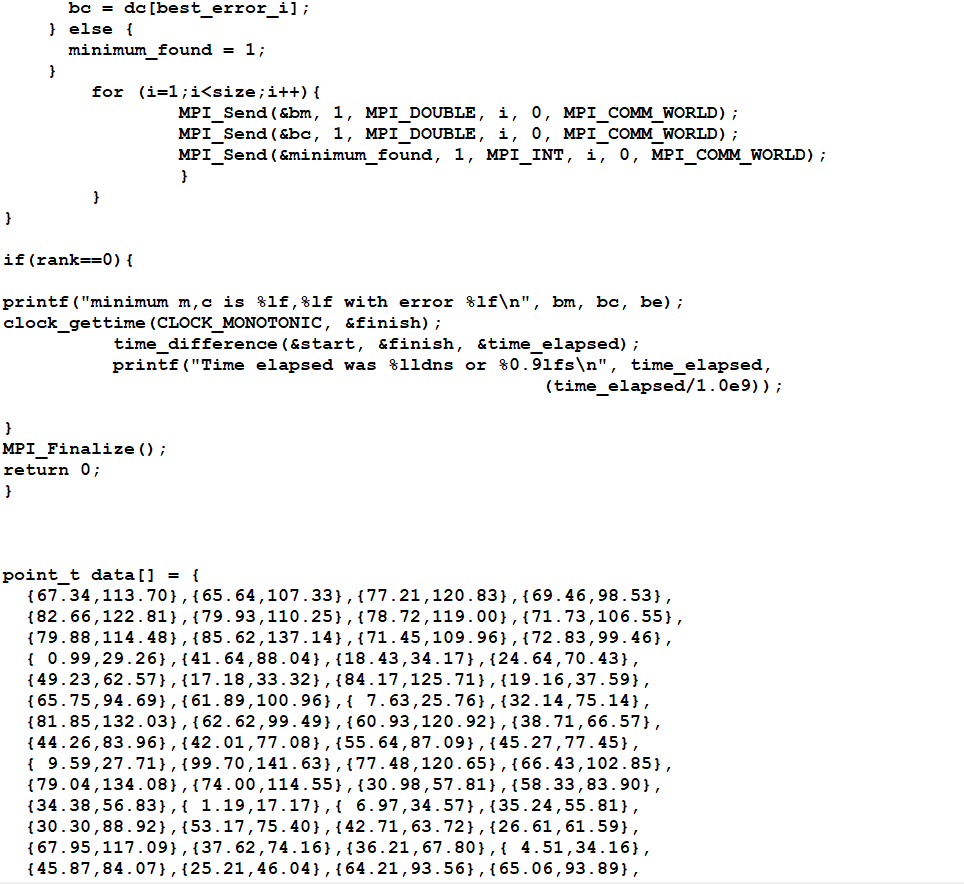


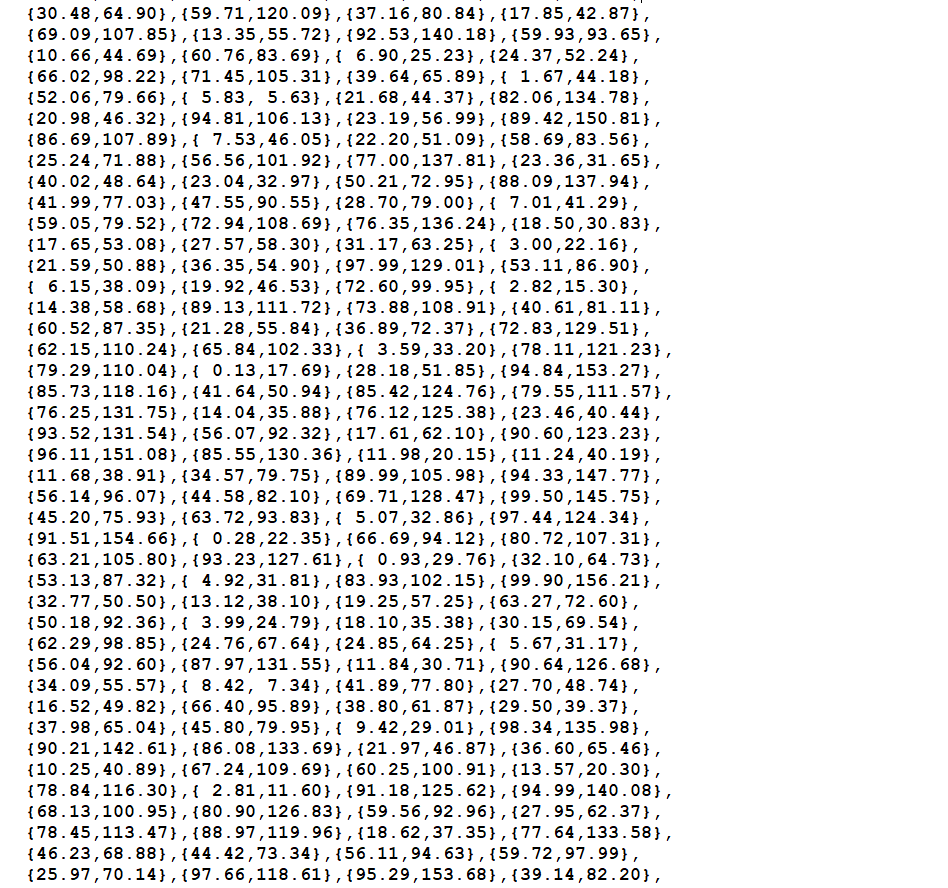
**Source code**

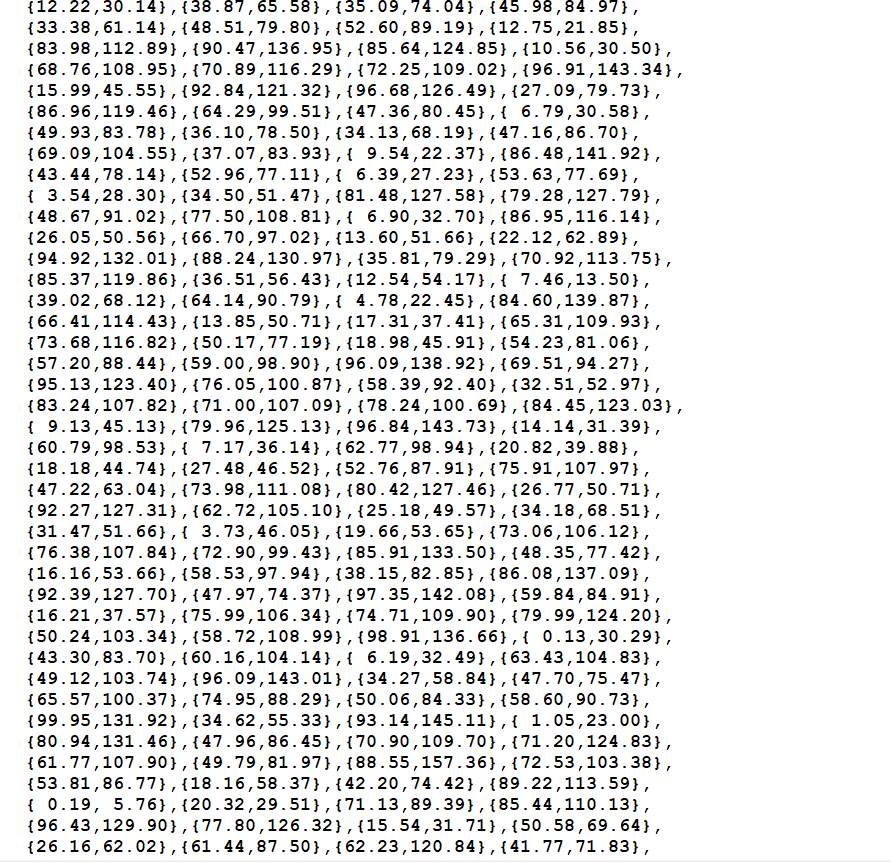
****

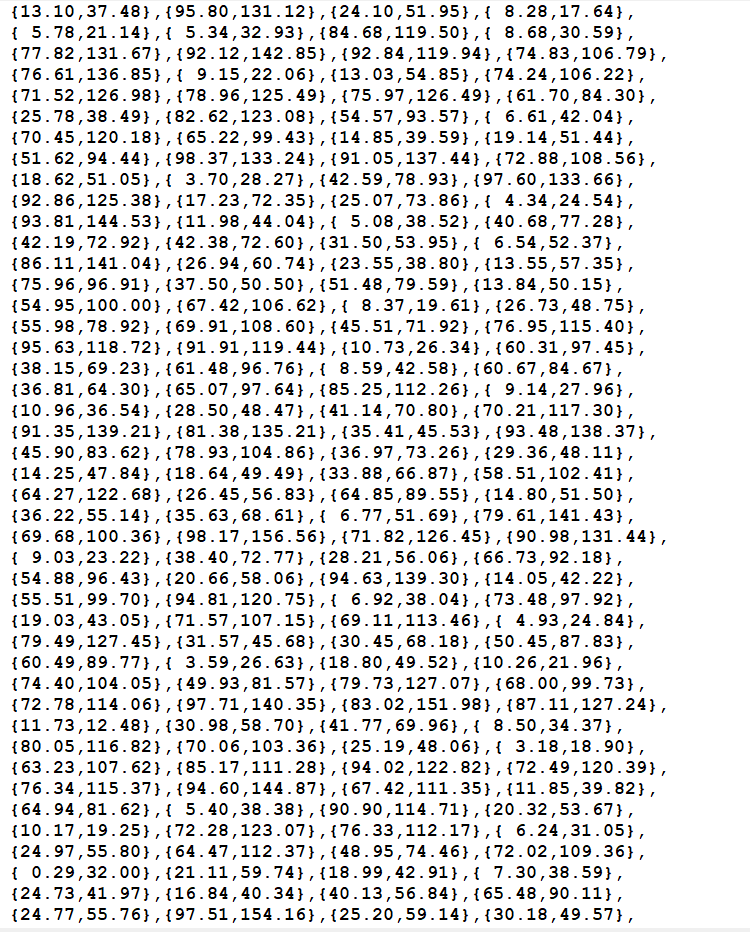
****

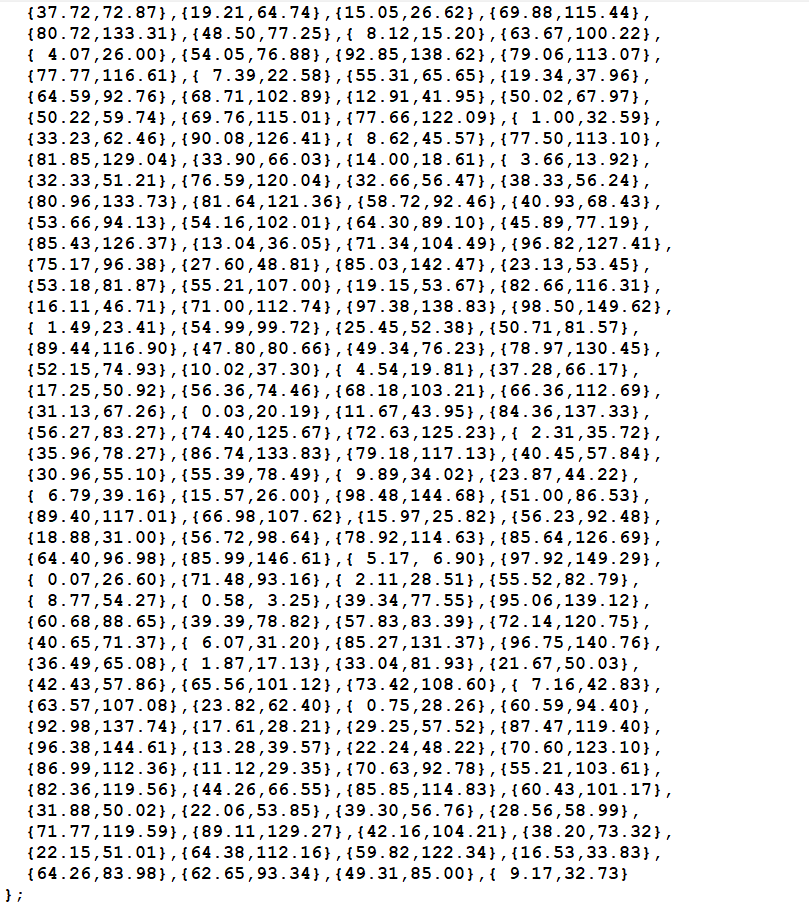
****

****

****

****

****

****

Calculating average time

